Particle Error Correction from Small Error Data for Japanese Learners

Kenji Imamura\(^1\), Kuniko Saito\(^{1,\dagger}\), Kugatsu Sadamitsu\(^\dagger\) and Hitoshi Nishikawa\(^\dagger\)

This paper shows how to correct the grammatical errors of Japanese particles made by Japanese learners. Our method is based on discriminative sequence conversion, which converts one sequence of words into another and corrects particle errors by substitution, insertion, or deletion. However, it is difficult to collect large learners’ corpora. We solve this problem with a discriminative learning framework that uses the following two methods. First, language model probabilities obtained from large, raw text corpora are combined with \(n\)-gram binary features obtained from learners’ corpora. This method is applied to measure the accuracy of Japanese sentences. Second, automatically generated pseudo-error sentences are added to learners’ corpora to enrich the corpora directly. Furthermore, we apply domain adaptation, in which the pseudo-error sentences (the source domain) are adapted to the real error sentences (the target domain). Experiments show that the recall rate is improved using both language model probabilities and \(n\)-gram binary features. Stable improvement is achieved using pseudo-error sentences with domain adaptation.

**Key Words:** Particle Error Correction, Language Models, Pseudo-Error Generation, Domain Adaptation, Discriminative Sequence Conversion

1 Introduction

Case markers in a sentence are represented by postpositional particles in Japanese. Incorrect usage of particles causes serious communication errors because readers cannot understand the content of sentences through the case markers. For example, it is unclear as to what must be deleted in the following sentence.

\begin{center}
\begin{tabular}{l}
mail \( o \) todoi tara sakuyo onegai-shi-masu \\
\end{tabular}
\end{center}

.mail ACC. arrive when delete please

“When \( \phi \) has arrived an e-mail, please delete it.”

If the accusative particle \( o \) is replaced by a nominative one \( ga \), it becomes clear that the writer wants the e-mail to be deleted (“When the e-mail has arrived, please delete it.”). Such particle errors frequently occur in sentences written by Japanese learners (c.f., Section 2).

\(^1\) NTT Media Intelligence Laboratories, NTT Corporation

\(^{\dagger}\) Presently with NTT Communications Corporation

* This article has been partially revised for better understanding of overseas readers.
Automatic error correction of texts written by Japanese learners is essential in education as well as business. For example, some Japanese corporations outsource the computer software development to foreign companies, especially China and India, using Japanese or English as the communication language. However, Japanese is a foreign language for most development companies, and texts such as e-mails can contain many errors that inhibit effective communication. Thus, automatic error correction is regarded as a key solution (Ohki, Oyama, Kitauchi, Suenaga, and Matsumoto 2011; Suenaga and Matsushima 2012).

This paper presents a method that can automatically correct Japanese particle errors. This task corresponds to preposition error correction in English. For English error correction, many studies employ classifiers, which select the appropriate prepositions/articles by restricting the error types to articles and frequent prepositions (Gamon 2010; Han, Tetreault, Lee, and Ha 2010; Rozovskaya and Roth 2011).

On the contrary, Mizumoto, Komachi, Nagata, and Matsumoto (2011) proposed machine-translation-based error correction. This approach can handle all error types by converting learners’ sentences into the correct version. Although the target of this paper is particle error, we employ a similar approach based on sequence conversion (Imamura, Izumi, Sadamitsu, Saito, Kobashikawa, and Masataki 2011) since it offers excellent scalability.

However, the machine-translation-based approach requires a sufficient number of parallel sentences, which consist of learners’ erroneous sentences and their manually corrected ones, to learn the models. However, collecting a large number of such pairs is difficult because of high costs. To avoid this problem, we propose the following two methods.

(1) Using Large Raw Text Corpora
   (Combination of Language Model Probabilities and Binary Features)
   Because one half of the parallel sentences are correct Japanese, they can be easily obtained from raw text corpora. Thus, we regard a large text corpus as a set of corrected sentences and incorporate it into the error corrector models. The text corpus is used for computing language model probabilities. The error corrector models are jointly optimized with binary features acquired from the parallel sentences by using discriminative learning. We expect that the error correction coverage will be improved because the degree of sentence correctness is measured by language model probabilities, even if the errors rarely appear in the parallel sentences.

(2) Expansion of the Parallel Sentences Using Pseudo-Error Sentences
   (and Application of Domain Adaptation)
   Since collecting learners’ erroneous sentences is not easy, we automatically generate pseudo-
error sentences, which mimic learners’ real errors, by applying error patterns to correct sentences (Rozovskaya and Roth 2010b). An additional training corpus consists of the pairs of pseudo-error sentences and their source sentences. Note that the pseudo-error sentences do not completely reflect the error distribution of learners’ sentences. Therefore, we further apply a domain adaptation technique that regards the pseudo-errors and real errors as the source and target domains, respectively. We expect stable improvement if the error distributions are partially different.

The remainder of this paper is organized as follows. In Section 2, we analyze errors in the sentences written by Japanese learners. Section 3 describes our error correction method and use of raw text corpora. Section 4 describes the expansion of parallel sentences using pseudo-error sentences and domain adaptation. In Section 5, we conduct experiments to confirm error correction accuracy. Section 6 introduces related studies, and Section 7 concludes this study.

2 Errors in Sentences Written by Japanese Learners

In order to analyze error types, we first collected erroneous examples written by Chinese native speakers who are learning Japanese.

Thirty seven subjects, who learnt Japanese while they are/were at engineering universities in Japan, participated in the experiment. The periods spent living in Japan ranged from six months to six years. We provided 80 English sentences obtained from Linux manuals and 24 figures to each subject (104 tasks in total), and they re-wrote the sentences in Japanese (Hereafter “learners’ sentences”). As a result, 2,770 learners’ sentences were collected. Each sentence was revised by Japanese native speakers (called the correct sentences). To make the revisions, a bare minimum of error correction was applied to the point at which the sentences become grammatically/pragmatically correct, while retaining the meaning. In other words, only fatal errors were corrected from the viewpoint of Japanese grammar.¹

2.1 Error Categorization and Distribution

We categorized the errors into three major types, grammatical, vocabulary, and surface form errors, and set sub-categories, as shown in Table 1.

The rewriters revised 2,171 of 2,770 sentences. The unchanged sentences included 559 sentences without errors and 40 sentence fragments. The following analyses were applied to the

¹ Vocabulary errors were corrected even if the sentences were grammatically acceptable.
Table 1 Error Categorization and Examples

<table>
<thead>
<tr>
<th>Category</th>
<th>Sub-category</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Particle/Auxiliary Verbs</td>
<td>shitsumon-*o/ni taiou-suru</td>
<td>“handle the question”</td>
</tr>
<tr>
<td></td>
<td>book-*aki-te/ake-te-kudasai</td>
<td>“please open the book”</td>
</tr>
<tr>
<td>Conjugation</td>
<td>20 MB-o koeru *dakara/-kara upload-shimasu</td>
<td>“upload it because it exceeds 20 MB”</td>
</tr>
<tr>
<td>Conjunction</td>
<td>sono/sore ika-no size-ni settei</td>
<td>“set the size to not greater than it”</td>
</tr>
<tr>
<td>Pronoun</td>
<td>*sono/sore ika-no size-ni settei</td>
<td>“set the size to not greater than it”</td>
</tr>
<tr>
<td>Grammatical Error</td>
<td></td>
<td>Interrogative, Word Order, Voice, Tense</td>
</tr>
<tr>
<td>Homonym</td>
<td>memory naizou (*内臓/内蔵)</td>
<td>“built-in memory”</td>
</tr>
<tr>
<td>Synonym</td>
<td>*kaisoku/jinsoku -ni shori-shimasu</td>
<td>“process it quickly”</td>
</tr>
<tr>
<td>Vocabulary Error</td>
<td></td>
<td>Confusion from Chinese</td>
</tr>
<tr>
<td>Transliteration</td>
<td>icon-o *kuriiku/kurikku -suru</td>
<td>“click the icon”</td>
</tr>
<tr>
<td>Pronunciation</td>
<td>shitsumon-ga *atarar/attara onegai-shimasu</td>
<td>“ask you if I have questions”</td>
</tr>
<tr>
<td>Typo</td>
<td>watashi-tachi (*立ち/達) -de yari-masu</td>
<td>“we do”</td>
</tr>
<tr>
<td>Surface Form Error</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

X and Y of **X/Y** in this table denote the error and its correction, respectively.

2,171 revised sentences.

The number of corrected errors was 4,916 words/phrases (2.26 per sentence). From the perspective of distribution by categories, most were grammatical errors (54%), followed by vocabulary errors (28%) and surface form errors (16%). The rest were compound errors. From the perspective of sub-categories, the most frequent errors were located in particles or auxiliary verbs (33%), followed by transliteration (11%) and synonyms (10%).

2.2 Analyses and Discussion

With regard to the most frequent errors, particle errors appear widely in texts written by non-native Japanese speakers regardless of their first language. This is because particles are special grammatical units in Japanese, and have various functions. For example, case particles specify the case of noun phrases such as nominative, accusative, and possessive. Topic particles work as topic markers, and can combine with case particles. There are other types such as conjunctive and sentence-end particles. Therefore, appropriate usage is difficult for most non-native speakers, and effective teaching should include correction of particle errors.

Since fallibility depends on the particle, we calculated the frequency of particle errors. Table 2 shows the top 10 particle errors.

Correction type in Table 2 denotes the edit operation needed to correct the errors: insertion

---

2 Particles are also used in Korean.
Table 2  Frequent Particle Errors (Top 10)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Error</th>
<th>Correct</th>
<th>Correction Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>wa/TOP.</td>
<td>ga/NOM.</td>
<td>SUB</td>
<td>117</td>
</tr>
<tr>
<td>2</td>
<td>o/ACC.</td>
<td>ga/NOM.</td>
<td>SUB</td>
<td>87</td>
</tr>
<tr>
<td>3</td>
<td>ϕ</td>
<td>no/POSS.</td>
<td>INS</td>
<td>70</td>
</tr>
<tr>
<td>4</td>
<td>o/ACC.</td>
<td>ni/DAT.</td>
<td>SUB</td>
<td>69</td>
</tr>
<tr>
<td>5</td>
<td>o/ACC.</td>
<td>ga/NOM.</td>
<td>SUB</td>
<td>66</td>
</tr>
<tr>
<td>6</td>
<td>ϕ</td>
<td>o/ACC.</td>
<td>INS</td>
<td>65</td>
</tr>
<tr>
<td>7</td>
<td>ga/NOM.</td>
<td>wa/TOP.</td>
<td>SUB</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>no/POSS.</td>
<td>ϕ</td>
<td>DEL</td>
<td>61</td>
</tr>
<tr>
<td>9</td>
<td>wa/TOP.</td>
<td>o/ACC.</td>
<td>SUB</td>
<td>54</td>
</tr>
<tr>
<td>10</td>
<td>ϕ</td>
<td>ni/DAT.</td>
<td>SUB</td>
<td>49</td>
</tr>
</tbody>
</table>

From the viewpoint of edit operation, substitutions were the most frequent (74%), particle omissions (INS) were 17%, and excess particles (DEL) were 9%.

The following are examples of substitution, insertion, and deletion. Rank 1 in Table 2 implies that the topic particle *wa* should appear only once in a sentence. If it appears twice, one of them should be replaced by a case particle *ga* (*mondai- wa/ga aru-toki- wa ... “if you have problem ...”). Rank 3 shows an example of insertion. Lack of the possessive particle *no* between a cardinal number and a noun was frequent (*futatsu-ϕ-no/le “two files”). Rank 8 shows an example of deletion. Many learners appended the possessive particle *no* after verbs or adjectives, which modify a noun (*chiisai-ϕ-e “small picture”).

The above analyses confirmed that particle error correction is effective for Japanese learners. These errors are corrected by applying the edit operations of substitution, insertion, and deletion.

### 3 Error Correction by Discriminative Sequence Conversion

This section describes error correction using discriminative sequence conversion. Our error correction method converts learners’ word sequences into correct sequences (sentences are segmented into words in advance by a morphological analyzer). The method is similar to phrase-based statistical machine translation (PBSMT), but has the following three differences: 1) it adopts conditional random fields, 2) it allows insertion and deletion, and 3) combines n-gram binary features and language model probabilities. Unlike the classification approach (e.g., (Suzuki...})
and Toutanova 2006)), the conversion approach can correct multiple errors of all types in a sentence.

3.1 Basic Procedure

We apply a morpheme conversion approach that converts the results of a speech recognizer into word sequences for language analyzer processing (Imamura et al. 2011). It corrects particle errors in the input sentences as follows.

- First, all modification candidates are obtained by referring to a phrase table. This table, called the confusion set (Rozovskaya and Roth 2010a) in the error correction task, stores pairs of incorrect and correct particles (Table 2). The candidates are packed into a lattice structure, called the phrase lattice (Figure 1). To deal with unchanged words, it also copies the input words and inserts them into the phrase lattice.

- Next, the best phrase sequence in the phrase lattice is identified on the basis of the conditional random fields (CRFs) (Lafferty, McCallum, and Pereira 2001). The Viterbi algorithm is used for decoding because error correction does not change the word order. Although the phrase lattice includes obviously ungrammatical sequences (e.g., a sequence in which identical particles are adjacent), we do not prune them, and search for the best sequence according to the model.

- While training, word alignment is carried out by dynamic programming matching. From

![Fig. 1 Example of Phrase Lattice (bold lines denote the correct sequence)](image)

Table 2 shows a part of the phrase table. As we will describe in Section 5.1, all words whose part-of-speech is **Particle** according to the ipadic-2.7.0 dictionary are processed.
the alignment results, the phrase table is constructed by acquiring particle errors, and the CRF models are trained using the alignment results as supervised data.\footnote{We implemented a CRF learner using Okazaki’s \liblbfgs as the optimizer. http://www.chokkan.org/software/liblbfgs/}

3.2 Insertion/Deletion

The error correction in this paper uses insertion and deletion, whereas phrase-based SMTs generally translate a sentence using only substitution. Since an insertion can be regarded as replacing an empty word with an actual word and deletion is the replacement of an actual word with an empty one, we treat these operations as substitution without distinction while learning/applyng the CRF models.

However, insertion is a high cost operation because it may occur at any location and can cause the lattice size to explode. To avoid this problem, we permit one only word insertion and only immediately after nouns. This restriction makes a few errors impossible to correct (e.g., it becomes impossible to insert the topic particle \textit{wa} immediately after the dative particle \textit{ni}).

Note that substitution can be represented as a sequence of insertion and deletion. In this paper, the supervised data for the CRF models are created while consecutive insertion and deletion are bundled as substitution. During error correction, multiple candidates consisting of the same surface forms exist in the phrase lattice; one is represented by substitution and the other is a sequence of insertion and substitution. However, substitution is selected in almost all cases by the learnt models.

3.3 Features

In this paper, we use mapping features and link features. The former measure the correspondence between input and output words (similar to the translation models of PBSMT), while the latter measure the fluency of the output word sequence (similar to language models). Figure 2 illustrates the features, and Table 3 shows the feature templates.

Many natural language processing (NLP) tasks that employ discriminative models (e.g., named entity recognition) use not only the targeted word but also its neighboring words as features. We employ the same approach. The focused phrase and its two neighboring words in the input are regarded as the window. The mapping features are defined as the pairs of the output phrase and uni-, bi-, and tri-grams in the window. All mapping features are binary.

Link features are detailed in the next section.
Table 3 Feature Templates

<table>
<thead>
<tr>
<th>Type</th>
<th>No.</th>
<th>Function Value</th>
<th>Template</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mapping</td>
<td>1</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i-2})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i-1})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Binary</td>
<td>$\delta(e_j, f_i)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i+1})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i+2})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i-2}, f_{i-1})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i-1}, f_i)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Binary</td>
<td>$\delta(e_j, f_i, f_{i+1})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i+1}, f_{i+2})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i-2}, f_{i-1}, f_i)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Binary</td>
<td>$\delta(e_j, f_{i-1}, f_i, f_{i+1})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Binary</td>
<td>$\delta(e_j, f_i, f_{i+1}, f_{i+2})$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>Binary</td>
<td>$\delta(e_j)$</td>
<td>$n$-gram binary feature</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Binary</td>
<td>$\delta(e_j, e_{j-1})$</td>
<td>$n$-gram binary feature</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Binary</td>
<td>$\delta(e_j, e_{j-1}, e_{j-2})$</td>
<td>$n$-gram binary feature</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Real</td>
<td>$\log(P(e_j</td>
<td>e_{j-1}, e_{j-2}))$</td>
</tr>
</tbody>
</table>

$\delta(\cdot)$ in the table denotes a binary function that returns 1 iff all arguments match the function, and 0 otherwise.

### 3.4 Using Raw Text Corpora and Incorporation into Link Features

The link features are important for the error correction task because the system has to judge output accuracy. The correct sentences can be easily obtained from raw text corpora. Using these corpora, we mix the two types of features, as follows, and optimize their weights in the CRF framework. We leverage a characteristic that discriminative models can handle features that depend on each other.

- **N-gram binary features**

  $N$-grams of the output words, from 1 to 3, are used as binary features. These are obtained
from the correct side of a training corpus (parallel sentences). Since the weights of individual features are optimized considering all features (including the mapping features), fine-tuning can be achieved. The accuracy becomes almost perfect on the training corpus. In other words, we can certainly correct errors in new texts if the same error patterns are present in the training corpus.

- **Language model probability**

  This is a logarithmic value (real value) of the $n$-gram probability of the output word sequence. One feature weight is assigned. The $n$-gram language model can be constructed from large text corpora, i.e., it does not require parallel sentences. The language model probabilities provide grammatical correctness scores, regardless of whether the errors in the new texts appeared in the training corpus or not.

  Incorporating binary and real features yields a rough approximation of generative models in semi-supervised conditional models (Suzuki, Isozaki, Carreras, and Collins 2009; Suzuki and Isozaki 2010). It can appropriately correct new sentences while maintaining high accuracy on the training corpus.

4 Expansion of Parallel Sentences Using Pseudo-error Sentences

The error corrector described in Section 3 requires parallel sentences, which correspond to bilingual sentences in machine translation. However, it is difficult to collect a sufficiently large set of such sentences. We resolve this problem by using pseudo-error sentences to expand them. In this section, we describe the generation of pseudo-error sentences and their application using domain adaptation.

4.1 Pseudo>Error Generation

As we described before, correct sentences, which are halves of the parallel sentences, can be easily acquired from raw text corpora. If we can generate errors that mimic learners’ sentences, we can obtain new parallel sentences.

We utilize the method of Rozovskaya and Roth (2010b). Concretely, when particles appear in the correct sentence, they are replaced by incorrect ones in a probabilistic manner by applying the phrase table (which stores the error patterns) in the opposite direction. The error generation probabilities are relative frequencies on the training corpus (i.e., the real error corpus). Namely,

\[ P_{\text{error}}(f|e) = \frac{C(f,e)}{C(e)}, \]  \hspace{1cm} (1)
where $P_{\text{error}}(f|e)$ denotes the error generation probability, $C(f, e)$ denotes the co-occurrence frequency of the correct particle $e$ and its error particle $f$ in the real error corpus, and $C(e)$ denotes the frequency of correct particle $e$ in the corpus.

The models are learnt using both real error corpus and pseudo-error corpus.

### 4.2 Domain Adaptation by Feature Augmentation

Although the error generation probabilities are computed from the real error corpus, the error distribution is not exactly the same. To fit the pseudo-errors to the real errors better, we apply a domain adaptation technique. Namely, we regard the pseudo-error corpus as the source domain and the real error corpus as the target domain, and models that fit the target domain are learnt.

In this paper, we use Daume (2007)'s feature augmentation method for domain adaptation. This method learns the target domain models by expanding its feature space, and has the same effect; that is, the models for the source domain are regarded as prior distribution for the target domain. In addition, it eliminates the need to change the learning algorithm.

We briefly review the feature augmentation. The feature space is segmented into three parts: common, source, and target. The features extracted from the source domain data ($D_s$) are deployed to the common and source spaces, and those from the target domain data ($D_t$) are deployed to the common and target spaces. Namely, the feature space is tripled (Figure 3).

Parameter estimation is carried out in the usual way on the above feature space. Consequently, the weights of the common features are emphasized if the features are consistent between the source and target. With regard to domain-dependent features (i.e., inconsistent between the source and target), the weights in the source or the target space are emphasized. With respect to the features that only appear in the source or target domain, the weights in the common and domain-dependent spaces are emphasized.

Figure 3 shows an example of feature augmentation. Here, we simplify the task to just asking whether the case particle $ga$ should be replaced with $o$ or left unchanged. We assume that the following three features are acquired from the source and target domain data (we assume template No. 11 in Table 3).

A) “Kinou ga riyou” appears in both source and target domains, and $ga$ is replaced with $o$.

B) “Data ga henkou” appears in both source and target domains. However, it is unchanged in the source domain and is replaced with $o$ in the target domain data.

C) “Kansuu ga jikkou” only appears in the source domain data.

When we estimate parameters on the above feature space, the weights of A) in the common space are emphasized because it is consistent between the domains. On the contrary, the weights
of B) in the common space become zero, and those in the source or target space are emphasized, depending on the correction, because the domains are inconsistent. Furthermore, the weights of C) in the common and source spaces are emphasized.

Error correction uses only features in the common and target spaces. The error distribution approaches the real error distribution because the weights of the features are optimized to the target domain. In addition, it becomes robust against new sentences because the common features acquired from the source domain can be used even when they do not appear in the target domain. From the examples in Figure 3, the features of C), which only appeared in the source domain data, can be used to enhance error correction.

5 Experiments

5.1 Experimental Settings

Target Particles: The target particles in this paper are words whose part-of-speech (POS) tag is Particle, according to the ipadic-2.7.0 morpheme dictionary. The dictionary includes not only case particles but also topic (focus), adverbial, conjunctive, sentence-ending, and parallel
particles (236 particles in total). Thirty-eight particles in the dictionary, which appear in learners’
corpus and were incorrectly used by the learners at least once, are targeted in the experiments. Learners’ Corpus (Real Error Corpus): The learners’ corpus used in the experiments con-
sists of 2,770 parallel sentences (104 tasks) collected in Section 2. From these sentences, only
particle errors were retained; the other errors were corrected by copying the corresponding parts
of the correct sentence. Therefore, the parallel sentences for the experiments contain only particle
errors. If just the POS tags of words were different (i.e., surface forms were identical) between
the pairs, we did not regard them as errors, and POS tags of the correct sentence were copied to
the learner’s sentence. The number of incorrect particles was 1,087 (8.0%) of 13,534. Note that
most particles did not need to be revised. The number of pair types of incorrect particles and
their correct ones was 132 (SUB: 95, INS:14, DEL: 23). All sentences in the experiments were
segmented and tagged by the Japanese morphological analyzer MeCab, using the ipadic-2.7.0
morpheme dictionary. The word information consisted of the surface form and its POS tag.
Language Model: This was constructed from Japanese Wikipedia articles on computers and
Japanese Linux manuals, 527,151 sentences in total. SRILM (Stolcke, Zheng, Wang, and Abrash
2011) was used to train a trigram model. During model construction, the modified Kneser-Ney
discounting and interpolation were used for backoff smoothing, and unknown unigrams were
retained as pseudo-word <unk>.
Pseudo-error Corpus: The pseudo-errors were generated using 10,000 sentences randomly
selected from the corpus for the language model. We changed the scaling factor of error generation
probabilities from 0.0 (i.e., no errors) to 2.0, while the relative frequency in the real error corpus
was taken as 1.0.
Evaluation Metrics: Five-fold cross-validation on the real error corpus was used. We used two
metrics as follows.
(1) Precision and recall rates, and F-measures of error correction by the systems. We compared
only surface forms of the words for score computation.
(2) Most particles did not need to be revised in this task. The system may excessively revise
particles that do not need to be corrected (over-correction). Therefore, we employ another
metric called relative improvement, which represents the difference between the number
of accurately corrected error particles and that of over-corrected particles (i.e., those that

5 In comparison with Suzuki and Toutanova (2006)’s study, which handled 10 case particles and topic particle
wa, we exclude dative particle e and include combined particle ni-yori instead of yori alone. In addition,
we handle topic particle wa as an individual word, whereas Suzuki and Toutanova (2006)’s study regarded a
sequence of case and topic particles as one word.
6 http://mecab.sourceforge.net/
did not need to be corrected). This is a practical metric because it denotes the number of particles that human rewriters do not need to revise after system correction. The relative improvement becomes zero if the system does not change any particle.

5.2 Experiment 1: Using Raw Text Corpora

First, we evaluate the effects of language model probability using raw text corpora in error correction. The following three methods were compared.

- **Proposed Method**: Incorporating the language model probability as a link feature with \( n \)-gram binary features.
- **\( N \)-gram Binary Features**: Only \( n \)-gram binary features are used as link features (i.e., the language model probability is not used).
- **Language Model Probability**: Only the language model probability is used as the link feature (i.e., \( n \)-gram binary features are not used).

Table 4 shows the results. In the table, \( \dagger \) denotes a significant difference between the proposed method and the \( n \)-gram binary feature method, and \( \S \) denotes a significant difference between the proposed method and the language model probability method (\( p < 0.05 \)).

Focusing on precision rates, the proposed and \( n \)-gram binary feature methods yielded the same accuracies, while the language model probability method yielded lower accuracy than the other methods. Focusing on recall rates, the proposed method had significantly better accuracy than the other two methods (from 9.9\% and 11.2\% to 18.9\%). This indicated that the proposed method had the highest F-measure. Comparing with the \( n \)-gram binary features and the language model probability methods, the recall rate of the language model probability method was slightly higher. Consequently, F-measure scores, in decreasing order, were the proposed method, the language model probability method, and the \( n \)-gram binary feature method.

However, focusing on relative improvements, the improvement of the language model probability method was degraded (i.e., over-correction occurred) even though the recall rate was higher.

<table>
<thead>
<tr>
<th>Link Features</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>Rel. Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed Method</td>
<td>50.2% (205/408)</td>
<td>18.9%(^\dagger) (205/1087)</td>
<td>0.274</td>
<td>+28(^\S) (205 - 177)</td>
</tr>
<tr>
<td>( N )-gram Binary Features</td>
<td>50.2% (108/215)</td>
<td>9.9% (108/1087)</td>
<td>0.166</td>
<td>+13 (108 - 95)</td>
</tr>
<tr>
<td>Language Model Probability</td>
<td>46.4% (122/263)</td>
<td>11.2% (122/1087)</td>
<td>0.181</td>
<td>-6 (122 - 128)</td>
</tr>
</tbody>
</table>

\(^7\) For the precision and recall rates, we used Pearson’s \( \chi^2 \) test. For the relative improvement, we used Student’s \( t \)-test.
than that of the $n$-gram binary feature method. This is because of the characteristic of this task; about 92% of the particles did not need modification. Improving the recall rate sometimes causes over-correction.

The proposed method had better relative improvement than the other two methods. However, the difference between the proposed and $n$-gram binary feature methods was not significant. We suppose that the $n$-gram binary features are especially effective in correcting certain errors, and have an advantage from the perspective of relative improvement. The proposed method improved the recall rate while maintaining the precision rate, and thus increased relative improvement. We conclude that combining the language model probability with the $n$-gram binary features is effective in improving error correction accuracy.

5.3 Experiment 2: Expansion of Parallel Sentences Using Pseudo-error Sentences

Next, we evaluate the effect of introducing pseudo-error sentences. The experiments were carried out by changing the usage of pseudo-error sentences; the link features are those proposed in Section 5.2.

Figure 4 plots the precision/recall curves for the following four combinations of training corpora and method. Note that each precision rate in Figure 4 was, at the specified recall rate, achieved by selecting corrected particles from high score corrections.

- **TRG**: The models were trained using only the real error corpus (baseline).
- **SRC**: The models were trained using only the pseudo-error corpus.

![Fig. 4 Recall-Precision Curve (the scaling factor of error generation probabilities is 1.0)](image-url)
ALL: The models were trained using the real error and pseudo-error corpora by simply adding them.

AUG: The proposed method. Feature augmentation was realized by regarding the pseudo-errors as the source domain and the real errors as the target domain.

The SRC case, which uses only the pseudo-error sentences, did not match the precision of TRG. The ALL case matched the precision of TRG at high recall rates. AUG, the proposed method, achieved higher precision than TRG at high recall rates. At the recall rate of 18%, the precision rate of AUG was 55.4%, while that of TRG was 50.5% (the significance was $p = 0.16$). Note that the precision rate of SRC was 35.6% at the recall rate of 18%, which is better than random correction.

Figure 5 shows the relative improvement of each method according to error generation probabilities. In this experiment, ALL achieved higher improvement than TRG at scaling factors of error generation probabilities ranging from 0.0 (no errors) to 0.6. Although the improvements were high, we have to control the error generation probability because the improvements in the SRC case fell as the scaling factor was raised. On the other hand, AUG achieved stable improvement regardless of the error generation probability. The relative improvement at a scaling factor of 1.0 was significantly improved because the improvement of TRG was +28 and that of AUG was +59 ($p < 0.05$). Thus, we can conclude that domain adaptation to pseudo-error sentences is the preferred approach.
5.4 Examples of Error Correction

In the second experiment (c.f., Section 5.3), the precision and recall rates of the proposed method (AUG) were 54.8% (210/383) and 19.3% (210/1,087), respectively, when the scaling factor of error generation probabilities was 1.0. The precision rate of 55% was not high enough because 45% of system modifications should be corrected again. However, some usages were clearly incorrect from the grammatical perspective, while others were acceptable because alternative particles can be accepted depending on the context. Accordingly, we carried out a subjective evaluation.

One hundred and seventy-three particles, which the system modified without matching the answer, were evaluated by one evaluator. Note that 151 of them were over-corrections, where the answers were not modified from learners’ sentences. We asked the evaluator whether the system modification would be acceptable, i.e., grammatically and semantically identical to the answer. As a result, 103 of the 173 particles were acceptable. Therefore, the precision rate including acceptable modification became 81.7% ((210 + 103)/383).

Table 5 shows examples of system corrections. Successful corrections were realized by substitution, insertion, and deletion. The acceptable examples included substitution from topic particle は to case particle が (No. 4), and dividing a compound noun by inserting the correct particle (No. 5). Unacceptable examples included the following: idiomatic expression was over-corrected (No. 7), the case particle for the passive voice was replaced with that for the active voice (No. 8), correction is impossible without knowledge of Linux’s free command (No. 10). No. 9 is similar to No. 4. However, it was judged unacceptable because “わたし-たち (we)” and “あたな (you)” are concord. Therefore, the same particles should be used. Features used in this paper only reflect local context of particles. Uncorrectable particle errors will persist unless global context features are introduced.

6 Related Studies

Particle error correction for Japanese learners has been researched for a long time. Recently, Suzuki and Toutanova (2006) used maximum entropy (ME) classifiers to restore particles (mainly case markers) omitted from sentences. They inserted the appropriate particle at the given position of the parsed and tagged sentences. Ohki et al. (2011) detected incorrect usage of particles in a learner’s sentence. They tagged and parsed the erroneous sentence and detected the incorrect usage (including lack of particles) by the support vector machines (SVMs), which used the features of neighboring words and dependency structure. Only detection was carried out.
Table 5  Examples of Error Correction by Proposed Method

<table>
<thead>
<tr>
<th>No.</th>
<th>In/Out</th>
<th>Sentence</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In</td>
<td>kore-o oshi-te, setsumei-gamen-*ga hyouji-shimasu.</td>
<td>Substitution</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>kore-o oshi-te, setsumei-gamen-α hyouji-shimasu.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“Push it and show the report screen.”</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>In</td>
<td>design-no henkou-no youkyuu -φ aru-node, yat-te itadaki-tai.</td>
<td>Insertion</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>design-no henkou-no youkyuu *ga aru-node, yat-te itadaki-tai.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“Please do it because there is a request to change the design.”</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>In</td>
<td>shin -φ system-kado-no-tame, ikou-ga hitsuyou-desu-node,</td>
<td>Deletion</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>shin -φ system-kado-no-tame, ikou-ga hitsuyou-desu-node,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“Migration is necessary to run the new system.”</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>In</td>
<td>juunbi- wa dekite-iru-no-deshou-ka?</td>
<td>Substitution</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>juunbi- ga dekite-iru-no-deshou-ka?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“Are you ready?”</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>In</td>
<td>mazu-wa design-no card -o sentaku-shite-kudasai.</td>
<td>Insertion,</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>mazu-wa design-no card -φ sentaku-shite-kudasai.</td>
<td>Division of</td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“First, please select a card of the designs.”</td>
<td>Compound Noun</td>
</tr>
<tr>
<td>6</td>
<td>In</td>
<td>futatsu-no suuji-no naka -φ, ookii suuji-o kaeshi-masu.</td>
<td>Insertion,</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>futatsu-no suuji-no naka -φ, ookii suuji-o kaeshi-masu.</td>
<td>The answer is de</td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“It returns the larger of two numbers.”</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>In/Out</th>
<th>Sentence</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>In</td>
<td>moushiwake-gozaimasen- ga, kono mail-o sakujo-shite-kudasai.</td>
<td>Substitution,</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>moushiwake-gozaimasen-φ node, kono mail-o sakujo-shite-kudasai.</td>
<td>Idiomatic</td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“I’m sorry, please delete this mail.”</td>
<td>Expression</td>
</tr>
<tr>
<td>8</td>
<td>In</td>
<td>“file-ga hason-shite imasu” to-iu message- ga hyouji-sare-masu.</td>
<td>Substitution,</td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“A message “the file is broken.” is displayed.”</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>In</td>
<td>watashi-tachi- wa shiken-o shitai-desu. anata- wa itsu junbi-deki-masu-ka?</td>
<td>Substitution,</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>watashi-tachi-wa shiken-o shitai-desu. anata- φ ga itsu junbi-deki-masu-ka?</td>
<td>Concord</td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“We want to test it. When are you ready?”</td>
<td>Expression</td>
</tr>
<tr>
<td>10</td>
<td>In</td>
<td>free-wa kernel- ga tsukat-ta buffer-o hyouji-suru.</td>
<td>Substitution,</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>free-wa kernel- φ tsukat-ta buffer-o hyouji-suru.</td>
<td>Necessity of</td>
</tr>
<tr>
<td></td>
<td>(Gloss)</td>
<td>“The free command displays buffers used by the kernel.”</td>
<td>Meaning</td>
</tr>
</tbody>
</table>

In English preposition/article correction, Han et al. (2010) corrected preposition errors using ME classifiers, where the features were neighboring words and head words acquired by parsers. Similarly, Gamon (2010) proposed a method to detect and correct preposition and article errors. The detection was based on ME classifiers, and the correction was based on decision trees.
Rozovskaya and Roth (2010a) corrected preposition errors using classifiers based on averaged perceptron models. These studies corrected errors by classifiers by restricting error types to those of particles, prepositions, and articles.

However, Mizumoto et al. (2011) proposed an error correction method that did not restrict the error type. They collected numerous pairs of learners’ sentences and their native correction from a social network service, and corrected errors by using a phrase-based SMT. Note that they did not segment learners’ sentences into words because they included errors. The erroneous sentences were segmented into characters. Compared to our study, the sequence conversion used in this paper is similar to that used in phrase-based SMT. Therefore, our method can handle all error types, i.e., it does not need to restrict errors to particles. However, our method assumes that the input sentences are segmented into words in advance. The accuracies of tagging and parsing might decrease when erroneous sentences are analyzed. We need to investigate word segmentation methods for erroneous sentences (Fujino, Mizumoto, Komachi, Nagata, and Matsumoto 2012).

From the perspective of correct sentence models, Suzuki and Toutanova (2006), Ohki et al. (2011), Han et al. (2010), Rozovskaya and Roth (2010a) modeled correct sentences using n-gram binary features. Gamon (2010), Mizumoto et al. (2011) modeled them as language model probabilities. In this paper, we combined both features and optimized them together with mapping features. As a consequence, the recall rates were improved.

From the perspective of learners’ sentence usage, all studies reported that the error correction accuracies improved by incorporating models that represent trends in learners’ errors with models of native speakers’ texts (Han et al. 2010; Gamon 2010; Rozovskaya and Roth 2010a; Kasahara, Fujino, Komachi, Nagata, and Matsumoto 2012). The method in this paper encompasses the above findings by using mapping features that represent learners’ error tendencies.

Rozovskaya and Roth (2010b) have proposed the use of pseudo-error sentences, which mimic learners’ sentences. They reported that error correction accuracy was improved by adding pseudo-error sentences, which have the same distribution as learners’ sentences. However, the best error generation method depended on the data (concretely, the learner’s native language), and error generation should be manually controlled. In this paper, we adjust the difference between pseudo-errors and real errors using the domain adaptation technique; hence, stable improvement is achieved.

Dahlmeier and Ng (2012) proposed a decoder that can correct multiple errors in a sentence. The decoder handles not only preposition/article errors but also typos, punctuation errors, and number agreement errors. The correction hypotheses are generated using classifiers and rules, which depend on the error type, and an input sentence is rewritten by hill climbing. The decoder
maintains multiple hypotheses while decoding in order to reduce the risk of falling into local optima. Our method maintains all hypotheses in the phrase lattice and uses the Viterbi algorithm to search for the best combination. The result is the optimal solution from the viewpoint of the models.

We described how most particles do not need to be corrected, and that improving the recall rate does not directly affect the relative improvement in this task. This phenomenon, called the imbalanced data problem, is a major problem when applying machine learning techniques to practical tasks (c.f., the survey paper (He and Garcia 2009)). Many solutions have been proposed to address this problem. For example, sampling methods aim to balance data by decreasing majority data or increasing minority data. The minimum Bayes risk methods learn models with different costs, depending on the majority and minority classification errors (in this task, correction errors). We need to investigate what methods can be applied to our task. Note that the pseudo-error sentences proposed in this paper are different from the over-sampling method because our purpose is to increase the training data without changing error distribution.

7 Conclusion

In this paper, we propose a method to correct particle errors in the sentences of Japanese learners. In the error correction task, it is difficult to collect sufficient number of pairs of learners’ and correct sentences. To avoid this problem, we first combine the binary features acquired from parallel sentences (small scale) and the language model probability derived from large, raw text corpora. By optimizing the above features via discriminative learning, we improve the recall rate of error correction. In addition, we generate pseudo-error sentences, which mimic learners’ sentences and add them to parallel sentences. By incorporating domain adaptation, stable improvement is achieved.

Our discriminative sequence conversion can handle not only particle errors but also all other error types. In the future, we aim to apply this method to as many other error types as possible.

Acknowledgement

A part of this study was presented at the 50th Annual Meeting of the Association for Computational Linguistics (Imamura, Saito, Sadamitsu, and Nishikawa 2012). The authors are grateful to Dr. Tomoko Izumi, who provided advice on the English translation of Japanese grammar. The authors appreciate helpful comments given by anonymous reviewers.
Reference


961
pp. 1049–1056, Sydney, Australia.


Kenji Inamura: graduated from Chiba University, Chiba, Japan, in 1985 and received the Ph.D. degree in engineering from the Nara Institute of Science and Technology, Nara, Japan, in 2004. He joined the Nippon Telegraph and Telephone Corporation (NTT) in 1985. He was with the NTT Software Corporation from 1995 to 1998 and the Advanced Telecommunication Research Institute International (ATR), Kyoto, Japan, from 2000 to 2006. He is currently with the NTT Media Intelligence Laboratories, Yokosuka, Japan, as a Senior Research Scientist. His research interests include machine translation, parsing, and predicate-argument structure analysis. He is a member of the ACL, the IEICE, the IPSJ, and the NLP.

Kuniko Saito: Senior Manager, Innovative Architecture Center, NTT Communications Corporation. She received the B.S. and M.S. degrees in chemistry from the University of Tokyo in 1996 and 1998, respectively. She joined NTT Information and Communication Systems Laboratories in 1998 and then moved to NTT Cyber Space Laboratories (now NTT Media Intelligence Laboratories). Her research focuses on part-of-speech tagging, named entity recognition, and term extraction. She is a member of the IPSJ and the NLP.

Kugatsu Sadamitsu: Researcher, Audio, Speech, and Language Media Project, NTT Media Intelligence Laboratories. He received the B.E., M.E., and Ph.D. degrees in engineering from Tsukuba University, Ibaraki, in 2004, 2006, and 2009, respectively. He joined NTT Cyber Space Laboratories (now NTT Media Intelligence Laboratories) in 2009. His current research interests include NLP and machine learning. He is a member of the Association for Natural Language Processing and the Information Processing Society of Japan (IPSJ).
Hitoshi Nishikawa: received B.A. (Policy Management) and M.M.G. from the Keio University in 2006 and 2008, respectively. He received his Dr. Eng. from Nara Institute of Science and Technology in 2013. He is currently a researcher at the NTT Media Intelligence Laboratories. His research interests include NLP, especially automatic summarization. He is a member of the Association for Computational Linguistics, the Association for Natural Language Processing, the Japanese Society for Artificial Intelligence and the Information Processing Society of Japan.

(Received April 18, 2012)
(Revised July 15, 2012)
(Accepted July 26, 2012)