Extraction of Useful Training Data for Article Correction Depending on User Inputs

Hokuto Ototake and Kenji Araki

Graduate School of Information Science and Technology, Hokkaido University,
Kita-14, Nishi-9, Kita-ku, Sapporo, 060-0814 Japan
{hokuto, araki}@media.eng.hokudai.ac.jp

Abstract. We present a method that extracts useful training data from a large corpus for our automatic article correction system. For Japanese learners of English, one of the most difficult problems is to use correct articles because they do not exist in Japanese. We previously proposed a system that detects and corrects article errors automatically. The system uses rules which are extracted automatically from conditions of word appearance, such as nouns and adjectives preceding them. Also more additional rules are generated from the recursively extracted rules by Inductive Learning which is proposed by us originally. However, since Inductive Learning needs too much time to make full use of a large corpus for training data, accuracy of detection and correction of errors appeared to be insufficient. Therefore we propose a method that extracts moderate amounts of training data from a large corpus depending on user input. We achieve a detecting article errors' F-measure of 0.70 and correcting F-measure of 0.58 which improved previous results for about 25%.

1 Introduction

An English noun phrase may have an article, such as “a”/“an” and “the.” It is one of the most difficult problems for Japanese (as well as other Asians) learners of English to use articles correctly. Lee[1] reported that native speakers of languages which do not have any articles often have difficulty in choosing appropriate English articles, and tend to underuse them. Also Kawai et al.[2] reported there are many article errors in English written by Japanese English learners.

Consequently, Japanese teachers of English have to correct a lot of article errors in writing. In many circumstances, they need to look up in a dictionary or check real text examples when correcting these errors because of the variety of situations requiring or not requiring articles. Thus it is laborious to correct these errors by hand.

In view of these circumstances, methods for detecting or correcting article errors automatically have been proposed in the past. Lee[1] uses combinations of article and noun phrase (NP) context features with a maximum entropy model to restore dropped articles. This method aims to restore dropped articles, so it does not cover substitution errors such as “a” erroneously replaced by “the.” Kawai
et al. [2] proposed a rule-based method which uses rules made by hand based on linguistic knowledge. To make these rules much effort and a lot of expenses are needed, and it is very difficult to achieve good coverage of article usage. Nagata et al. [3] and Izumi et al. [4] proposed statistical models for detecting article errors. These methods do not need to create dictionaries or rules for detecting errors, but have problems using complex contexts. Also, they do not handle unusual usage of articles well.

To solve the problems with the existing methods, we have proposed a system [5] for correcting article errors automatically using Inductive Learning (IL) [6]. This system uses rules involving a combination of an article and feature slots constructed based on the NP head, adjective(s) in the context, and so on. Rules are extracted from corpora of error free text. Also, additional abstract rules are generated recursively from the extracted rules by IL. Applying the resultant rules to user inputs, the system detects and corrects article errors. This has the advantage of making rules automatically and finding context factors relating to article selection by abstracting rules. However, it is impossible for the system to make full use of a large corpus of training data, because the number of steps in recursive IL grows exponentially. For this reason, the system's accuracy in detecting and correcting errors appeared to be insufficient.

In this paper, we propose a user input dependent method that extracts moderate amounts of training data from a large corpus. This method aims to both improve accuracy and to speed up the error correction process, by using the same corpus more efficiently.

The rest of the paper is organized as follows: In section 2, we describe the algorithm for correcting article errors based on [5]. Section 3 presents the proposed method that extracts rules from a corpus. In section 4, we evaluate the method and section 5 has our conclusions.

2 Correction algorithm

2.1 Extraction of feature slots and rules

First, we describe the contents of the rules our system uses for correcting article errors. Consulting literature concerning usage of articles [7, 8], we define feature slots for the context in a sentence. A rule consists of a combination of an article and such feature slots. They are extracted from parsing sentences automatically. We use The Stanford Parser [9] to parse sentences. As an example:

(i) This is the only book which I bought yesterday.

The feature slots and rule extracted from “book” in sentence (i) is as shown in Fig. 1. Feature slots consists of three categories, Target, Preceding and Following. Target category has context information about a target noun. If a target is a compound noun such as “tennis player”, “tennis player” is put into the Nouns slot and “player” is put into Head. The Preceding category means information about modifiers preceding the target noun, such as an adjective or adverb. The Following category contains information about modifiers
Fig. 1. An example of a rule.

following the target noun, such as prepositions, infinitive and relative clauses. In sentence (i), since the target noun “book” is modified by a following relative clause “which”, this information is put into the Relative slots.

A requirement when applying rules to new text is that the rule feature slots and the target noun’s context must agree. If the slot element is “-”, it counts as agreeing with any other element.

2.2 Inductive Learning

In this paper, Inductive Learning is used as a method for discovering inherent regularities from actual examples[6]. The actual examples in our case are the feature slots extracted from training data. In the IL process, abstract rules are generated one after another by extracting common and different elements recursively from comparison of two rules’ feature slots. This process aims to generate rules which have only context elements dominating article selection.

Fig. 2 shows an example of a new rule being generated by IL. In Fig. 2, the two upper rules are extracted from sentence (i) and the sentence (ii) below. The new rule is generated by IL from the two upper rules.

(ii) Bobby is the only child in his family.

The two upper rules have many common slot elements: all of Preceding, a portion of Target and the article. The other elements differ. IL thus generates a new rule which has these common slot elements and abstract elements at the differing slots. In Fig. 2, elements with “*” similarly to “-” means the slot is allowed to agree with any element.

If the only requirement for IL was that two rules must have common slot elements, many rules abstracting too much would be generated. Thus, we add
* For clarity, the Following category is not included.

Fig. 2. An example of when the IL process generates a new abstract rule.

the requirements that the article elements of the rules must agree and that Target Head or Preceding Modifier must also agree. IL then generates rules recursively until no rules meeting these requirements remain.

3 Process

In this paper, we use a system consisting of the algorithm described in section 2 and the extraction method described below to correct article errors. Fig. 3 shows the system flow.

When the system gets an English sentence including article errors to correct as user input, it first parses the sentence with The Stanford Parser[9] and extracts feature slots. Using the resultant feature slots as a query, the system extracts training data from a corpus. The system corrects article errors by the correction algorithm using the resultant training data. In the correction algorithm, the system calculates scores of rules to rank them by reliability.

3.1 Extraction of training data

It is impossible for the system to generate rules from the whole of a large corpus (say over a million words) because of the exponential growth of the recursive
IL. Therefore, we try to extract moderate amounts of training data from a large corpus using the feature slots of the user input sentence. Fig. 4 shows an example of extracting training data for the target noun “practice” in the following sentence (iii).

(iii) By the common practice of paying with plastic, ...

When a user sentence is input, queries based on the noun phrases and their contexts are generated for each noun. The query patterns are as follows, in priority order:

1. preposition + premodifier + noun
2. premodifier + noun
3. noun
4. head
5. premodifier

preposition means a preposition preceding a target noun if it belongs to a prepositional phrase, and the premodifier is a preceding modifier. Since the difference between the noun and head are only different for compound nouns, these
may be the same. In the example sentence (iii), the head and noun are "practice", the premodifier is the adjective "common", and the preposition is "by".

The generated queries are used with a full-text searching system in order of priority to extract training data from a large corpus. We use HyperEstraier[10], an open-source full-text search system. As a query consisting of several words is received by the full-text search system, it extracts documents that include the query words in order. Ranked by priority order, it is possible to extract training data in descending order of relevance to the target NP context, which helps the system to generate effective rules. However, in case of multiple word queries, it may be the case that no document is extracted. In such cases, we go on to try single word queries, the head noun and premodifier. The premodifier is used in cases where there is not enough extracted training data from using the head noun, such as for very rare nouns.

3.2 Calculation of scores for rules

After extracting and generating rules from the training data, the system tries to calculate reliability scores for each rule using documents extracted by the method described in section 3.1. The amount of training data extracted is limited by IL processing time. Basically, the system uses about 200 sentences of training data. However, by calculating reliability scores for the rules using extracted documents other than the original training data, we try to improve the reliability of rules.

Now we define "NA (number of times applied)" and "NAC (number of times applied correctly)". NA means the number of times a rule is applicable (context matches feature slots) in the documents. NAC means the number of times the rule is applicable and the article used is the same as the one suggested by the rule. The score of the rules is defined by the following formula:

\[
\text{score} = \frac{NAC}{NA}
\]  

(1)

When correcting article errors, the system prefers rules with high specificity levels and high scores. Specificity level means the proportion of non-wild card elements (i.e., other than "*" and "**") to all feature slots. Additionally, we also set two other parameters in the system. The first one is the threshold \( \theta \). The system uses only rules with scores greater than or equal to \( \theta \). Another parameter is the threshold \( n \), which means the maximum number of rules to apply to the same NP.

The system selects rules in order of specificity (only rules score \( \geq \theta \)). If rules have the same specificity, the one with the highest score is used. When several rules are applicable but suggest different articles, the system cannot correct the error, but it can detect an error if the used article is not suggested by any rule.

4 Experiments

In this section, we describe the evaluation experiment and a comparison with our previous system[5].
4.1 Corpus for training data

We use the Reuters Corpus[11] as training data, which includes about 256 million words (the number of unique words is about 240,000). In our previous system[5], it was necessary to prepare rules in advance. Extracting rules and training for about one week on a server with a Core 2 Quad Q6600 processor and 3GB memory, the number of words that could be used as training data was about 170,000 (about 4,200 unique words). In the proposed system, as previously indicated, new training data is extracted automatically from the whole corpus for each user input. The average time taken by the previous and the proposed system to return their judgements are 1 second and 18 seconds per sentence respectively. Although the previous system needs only 1 second, it uses only 0.07% of the corpus as training data.

4.2 Test data

The test data consists of 9 files from the Reuters Corpus. We have replaced the articles with blanks and also added blanks before all noun phrases. Then two Japanese college science students were asked to fill in the blanks with articles or leave their blanks if not needed. The test data is not part of the training. The number of words is 1,586 (including 249 articles) and 121 article errors are included in the test data after being filled in by the students.

4.3 Experimental procedures

First, we set the parameters for the system. In order to set the threshold \(\theta\), we randomly extracted 100 sentences from the EDR corpus[12], and applied the system to them with the threshold \(n = \infty\). Because these sentences do not include article errors, the rules should be applied without detecting any. Fig. 5 shows the relationship between threshold \(\theta\) and noun phrase coverage (the proportion of noun phrases for which at least one rule matches the phrase context). Fig. 6 shows the relationship between \(\theta\) and the false detection rate. In Fig. 5 and Fig. 6, the false detection rate is almost constant and the noun phrase coverage is high for \(\theta = 0\) to 0.5. We concluded that increasing \(\theta\) had a positive effect on precision and set \(\theta\) to 0.5.

Finally, we detected and corrected article errors in the test data. Threshold \(n\) was set to 1 for "recall-oriented" and \(\infty\) for "precision-oriented."

4.4 Evaluation method

We use recall and precision to evaluate the system. \(R_d\) (Recall) and \(P_d\) (Precision) for error detection are defined as follows:

\[
R_d = \frac{\text{the number of article errors detected correctly}}{\text{the number of article errors in the test data}}
\]  

(2)

\[
P_d = \frac{\text{the number of article errors detected correctly}}{\text{the number of detected article errors}}
\]  

(3)
Additionally, $R_c$ (Recall) and $P_c$ (Precision) for error correction are defined as follows:

$$R_c = \frac{\text{the number of article errors corrected correctly}}{\text{the number of article errors in the test data}} \quad (4)$$

$$P_c = \frac{\text{the number of article errors corrected correctly}}{\text{the number of detected article errors}} \quad (5)$$

We also define $F$ (F-measure[13]) in both error detection and correction as follows:

$$F = \frac{2 \cdot P \cdot R}{P + R} \quad (6)$$

### 4.5 Results and discussion

Table 1 shows the results of detecting errors. The performance of the proposed system is better than the previous system. By comparison between $n = 1$ and
Table 1. Results of error detection.

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed system ($n = 1$)</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
</tr>
<tr>
<td>Proposed system ($n = \infty$)</td>
<td>0.42</td>
<td>0.76</td>
<td>0.54</td>
</tr>
<tr>
<td>Previous system [5]</td>
<td>0.35</td>
<td>0.67</td>
<td>0.46</td>
</tr>
</tbody>
</table>

Table 2. Results of error correction.

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Precision</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed system ($n = 1$)</td>
<td>0.59</td>
<td>0.58</td>
<td>0.58</td>
</tr>
<tr>
<td>Proposed system ($n = \infty$)</td>
<td>0.29</td>
<td>0.52</td>
<td>0.37</td>
</tr>
<tr>
<td>Previous system [5]</td>
<td>0.22</td>
<td>0.43</td>
<td>0.29</td>
</tr>
</tbody>
</table>

$n = \infty$ in the proposed system, $n = 1$ gives a better $F$-measure than $n = \infty$, but for precision the result is the opposite. Thus, the threshold $n$ does indeed affect the trade-off between recall and precision.

Table 2 shows the results of correcting errors. As in the results for detection, the performance of the proposed system is better than the previous system. However, both recall and precision of the proposed system for $n = \infty$ are worse than for $n = 1$. Increasing the number of rules applied at the same time also increases the number of correction suggestions to choose from, lowering recall without increasing precision.

Although the proposed system had better performance than the previous system, about 30% of the article errors still cannot be detected by the system for $n = 1$. Most, about 66%, of the errors that are not detected have "the" as the correct article. Since the usage of "the" is affected by prior context or sentence meaning, the present system cannot deal with such errors. To solve this problem, it is necessary to add feature slots for prior context. In the case of the remaining 40%, there is no rule that is suitable for them. Most of these errors include rare nouns or proper nouns such as a company names, and we believe that the system could detect such errors by increasing the size of the training data.

On the other hand, about 30% of the detected errors are false detections using $n = 1$. One of the causes of these is the effect of context, the same as for undetected errors. We believe that another cause is the combination of determinative adjectives and common nouns. Queries made by the proposed system for extracting training data include adjectives as modifiers, but the priority is the lowest. As a result, the system may not generate rules including the adjective. To solve this problem, we need to improve the algorithm that makes extraction queries.
5 Conclusions

In this paper, we proposed a method that extracts the most appropriate training data from a large corpus for our automatic article correction system. The experiment results showed that the proposed system performed better than our previous system. Both precision and recall of both detection and correction of errors were improved by selecting appropriate parts of the whole large corpus of training data efficiently, and the processing time was reduced very significantly.

However, the detection and correction of errors task, precision is more important than recall, because false detection or correction of a system would confuse the user. Therefore precision improvement is the top priority for the future. We believe that the proposed method for extracting training data can be improved when it comes to creating queries. We will also try to evaluate the system with a larger corpus.

References