English Article Errors Correction

jjjj222

1 Overview

This project is about grammatical error correction, and the focus here is article ("a", "an", "the") errors. We use classification approach. And the experiment is 10-fold cross validation on NUCLE dataset. The precision, recall, and f1 value are 12.8%, 34.7%, and 0.19.

2 Introduction

English is the most widely used language for communication, and it is very important to use it correctly. However, it's not easy to get everything right — even a native speaker would make mistakes at times. This problem is especially struggling for English learners, because most of the time they don't even know they made a mistake. Imagine if there is a NLP tool which can help us find these errors, performing tasks such as spell checking, grammatical error correction, word choice, and even paraphrase, it would not only save us a lot of time but also improve the quality of communication. This project focuses on grammatical error correction, because it's the most common mistakes for an English learner, and it's very hard for them to notice these mistakes by themselves.

There are many contests over the past few years on grammatical error correction, such as HOO 2011 and 2012 [1, 2], and CoNLL 2013 and 2014 [3, 4] shared taskes. These tasks are similar. They provided some documents with human annotations, indicating errors and their corrections, and called for tools to perform these annotations automatically. The difference between these contests is the target errors to correct; some have more, and some have less. However, one type of the error they all have is article error.

Article error is the most common error in written English. It's very subtle, sometimes simply idiomatic, to choose one but not the other. And it's especially true for English learners with the language background that does not use articles. This project aims to build a system that can automatically correct article errors.

3 Problem Definition

This work aims to identify article errors (the mis-use of "a," "an", and "the"). Article errors can be roughly categorized into the 3 different types:

1. Deletion errors: There should be an article but the writer forgot to add it. e.g. She is [the] fastest runner.
2. Substitution errors: The writer used an incorrect article. e.g. They play ~~the~~ [an] important role in our life.
3. Insertion errors: The writer added a redundent article. e.g. I live in ~~the~~ Asia.

Note that in this work we do not distinguish "a" and "an", which means the mis-use of a/an is not included here (e.g. It's ~~a~~ [an] apple). The reason is that they can be easily fixed by post-processing with simple rules.

4 Usage

There are 2 steps to use this program. The first step is to preprocess input file: separate text and annotation, tokenize text, apply PoS taggings, and do dependency parsing. And the second step is running the correcter. The details to use them are listed below.

4.1 Preprocessing

The input of this program is an SGML file. The script to preprocess SGML files are provided by NUCLE dataset [5]. To use the script, type the following commands under the project directory:

```
cd data/nucle3.2/scripts
python preprocess.py -o <sgml_file> <conll_file> <ann_file> <m2_file>
```

The "`<sgml_file>`" is input file; the "`<conll_file>`" and "`<ann_file>`", which contain text and annotation respectively, are intermedate files for the next stage. The "`<m2_file>`", which is for the performance evaluation in CoNLL 2014 shared task [4], is not used in this project.

4.2 Run Correcter

To run the correcter, type the following commands.

```
cd src
python AutoCorrect.py <conll_file> <ann_file> <log_file>
```

The "`<conll_file>`" and "`<ann_file>`" files are from previous step. And the results will be written to "`<log_file>`". Note that for conciseness we don't print out the real correction, because it's not the focus here (and can be easily done). The "`<log_file>`" will include statistic information only.

5 Approach and Implementation

Besides rule-based approaches, the most common methods for grammatical error correction problem are classification and statistical machine translation [6]. Statistical machine translation (SMT) approach models a correction problem as a translation problem, where the source language is "bad English" and the target language is "good English", while classification approach aims to classify whether a word in sentence is "correct" or "incorrect". We use the classification approach in this project.

The classification approach is based on the pipeline architecture (Figure 1). The adventage of this approach is that each type of error can be resolved in each stage separately. The reason we choose this approach is because our problem is a single-point-fix problem, and it fits in this paradigm. On the other hand, the SMT approaches try to

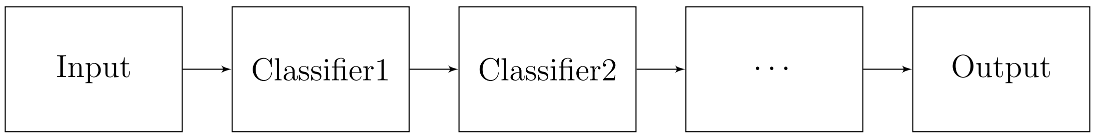

Figure 1: The pipeline architecture.

process a sentence as a whole. It can fix multiple problems at the same time, but it's not as accurate as classifier and requires a lot of parallel sentences, and it doesn't not easy to include domain specific knowledge.

The basic flow of our approach is as follows: 1) preprocessing, 2) candidates identification, 3) feature extraction, 4) correction.

5.1 Preprocessing

The input of this program is an SGML file, and it should include both texts and annotations. In the preprocessing, the texts and the annotations will be separated. The texts will be tokenized by NLTK [7] tokenizer, and then be parsed by Stanford Parser [8]. The tokenized corpus will be save to "`.conll`" file with format as showed in Figure 2. On the other hand, the annotation will be transferred from character-level to word-level, and be stored in "`.ann`" file (Figure 3).

```
11  3   5   0    They         PRP     1    nsubj       (ROOT(S(NP*)
11  3   5   1    play         VBP     -1   root        (VP*
11  3   5   2    the          DT      4    det         (NP(NP*
11  3   5   3    important    JJ      4    amod        *
11  3   5   4    role         NN      1    dobj        *)
11  3   5   5    in           IN      4    prep        (PP*
11  3   5   6    our          PRP\$   7    poss        (NP*
11  3   5   7    life         NN      5    pobj        *))
11  3   5   8    which        WDT     12   nsubjpass   (SBAR(WHNP*)
11  3   5   9    can          MD      12   aux         (S(VP*
11  3   5   10   not          RB      12   neg         *
11  3   5   11   be           VB      12   auxpass     (VP*
11  3   5   12   substituted  VBN     4    rcmod       (VP*)))))))
11  3   5   13   .            .       -    -           *))
```

Figure 2: The preprocessed 5th sentence "They play ~~the~~ [an] important role in our life which can not be substituted." in 3rd paratraph in document number 11.

```
    <ANNOTATION>
        <MISTAKE nid="11" pid="3" sid="5" start_token="2" end_token="3">
        <TYPE>ArtOrDet</TYPE>
        <CORRECTION>an</CORRECTION>
        </MISTAKE>
    </ANNOTATION>
```

Figure 3: The annotation for the error in the fifth sentence "They play ~~the~~ [an] important role in our life which can not be substituted." in 3rd paratraph in document number 11.

5.2 Candidate Identification

For substitution and insertion errors, the candidates are obvious — every "a", "an", and "the" in the text. For deletion errors, any place between 2 token could be a candidate. However, the only valid place to add an article is at the front of a noun phrase (NP). For examaple. In sentence "They play ~~the~~ [an] important role in our life", the valid places are "they", "important role", "our life".

Since we can rule out articles which are not at the front of a noun phrase, we only have to check every noun phrase.

5.3 Feature Extraction

For every candidate location, we extract some features that we think might be useful for this problem. There are total 17 features, as listed in Table 1. Note that the feature "`current-det`" can only be used on dataset with annotations. Because if this feature is used on corpus without error, it will soley decide the classification result.

Feature	**Description**	**Example**
head-word	Head word of NP	role
first-word	First word in NP	important
prev-word	First word before NP	play
prev-word-2	Second word before NP	they
next-word	First word after NP	in
next-word-2	Second word after NP	our
head-tag	POS of the head word of NP	NN
first-tag	POS of the first word in NP	JJ
prev-tag	POS of first word before NP	VBP
prev-tag-2	POS of second word before NP	PRP
next-tag	POS of first word after NP	IN
next-tag-2	POS of Second word after NP	PRP$
head-dprel	Dependency tag of head word of NP	dobj
current-det	Current determiner	the
head-parent	Parent of head word in dependency tree	play
prev-verb	First verb before NP	play
next-verb	First verb after NP	<NULL>

Table 1: Features of NP "the important role" in sentence "They play ~~the~~ [an] important role in our life."

5.4 Correction

After features of a noun phras in the training set are calculated, these features, along with the correct article for the noun phrase, will be seen as a case used to train a classifier. The correct article for a noun phrase is:

1. Noun phrase without annotation: current article.

2. Noun phrase with annotation: the correction.

Once the training is done, the classifier could be use to classify the features from the test set. If the result of a noun phrase is different than the current article, then a new correction will be created.

6 Experimental Results

6.1 Corpus

The dataset used in this project is NUS Corpus of Learner English (NUCLE) [5] release 3.2 version. It is a collection of 1,397 essays written by students at the National University of Singapore (NUS) who are non-native speakers of English. For each grammatical error instance, the start and end character offsets of the erroneous text span are marked, and the error type and the correction string are provided. The error annotations are saved as stand-off annotations, in SGML format. There are total 28 error types, and 6,640 (14.8%) out of all 44,912 annotations are ArtOrDet (article or determiner). More details are in Table 2.

Data	**Number**
essay	1,397
sentence	57,151
word token	1,161,567
error type	28
all error	44,912
ArtOrDet error	6,640 (14.8%)
Wci error	5,305 (11.8%)
Rloc error	4,703 (10.5%)

Table 2: Statistics of NUCLE dataset

6.2 Metrics

Since this project focuses only on article error, all errors except ArtOrDet will be excluded in evaluation. In addition, if an error is about determiner but not article, it will not be counted as well. The total number of target errors is 5,416 , and the number of candidates is 259,851 in the experiment.

We use precision, recall, and f1 value as metrics here because there is no “False Negative” in this problem. All proposed corrections will be gathered and compared to the golden ones. Those have a match with a golden correction on both location and correction will be counted as “True Positive”, golden corrections without any match are “False Negative”, and proposed corrections without any match are “False Positive”. If a proposed correction get the location right but with wrong correction, or vice versa, it would be counted as both FP and FN.

All results are average values calcuated through 10-fold corss validation. Note that we separate cases by documents instead of sentences, so all sentences in the same document would be in the same set as well.

6.3 Results

We run 3 different classifiers: Naive Bayes, Decision tree, and MaxEnt. And each of them has 2 addiitonal options: “no current-det” and “dup correction”.

The option “no current-det” means that it run without the “current-det” features. The reason of this option is because, by removing the “current-det” feature, this error correction problem could be seen as a language model problem, and then the classifiers can be trained on corpus without annotations, which is a huge adventage due to the limited number of parallel dataset.

And the option “dup correction” means that during the training, cases with annotation (the current article is not the same as expected one) will be duplicated 10 times in the training set. It’s for balancing the output class in training set while using “current-det” feature, because the number of candidates which need a modification is much less than those which need not.

The highest f1 value is 0.19 by using MaxEnt classifier with duplicated-correction option. And the highest precision is 22% by using MaxEnt with the current-det feature disabled. Hoever, the recall in this case is very low. The details are in Table 3.

Classifier	Option	Precision	Recall	F1
Naive		8.9%	27.6%	0.13
	no current-det	4.7%	51.0%	0.09
	dup correction	6.4%	45.1%	0.11
Decision Tree		11.9%	8.3%	0.10
	no current-det	3.9%	49.9%	0.07
	dup correction	7.9%	24.7%	0.12
MaxEnt		22.0%	3.2%	0.06
	no current-det	7.1%	56.9%	0.13
	dup correction	12.8%	34.7%	0.19

Table 3: The precision, recall, and f1 value of different classifiers.

6.4 Discussion

The option “no current-det” lowers precisions and increase the recall in all cases. It’s expected because this features strongly correlated to the correct article (because most of the usage are correct). We like the idea of transforming the correction problem into a language model problem. However, it seems that more training data are required for this option. On the other hand, using “dup correction” to balance the number of case that requires correctoin seems beneficial. It increases the f1 value of MaxEnt from 0.06 to 0.19.

However, it’s still not very good in comparsion with some state-of-the-art results (even though the evaluation method is different). Sun et al. [9] claimed they reached 0.381 f1 value on CoNLL-2013 data set by convolutional neural network, and outperformed the best system (0.334) at the time. And Garimella [10] showed, by using recurrent neural network with bidirectional LSTM, his work has a maximum precision of 51%, recall of 34%, and F0.5 score of 0.47.

Even though neural networks seems very powerful, there are still several potential methods that can be used to boost the performace under current architecture. But due to the limitation of time, we are unable to implement them. First of all, by adding more features and perform feature selection, the overall perforamce should be able to further increase. Unfortunately, it takes us more than 24 hours to run a 10-fold cross validation for MaxEnt classifier, so we don’t have enough resources to do that. Second, because some choices of the article seems to depend on context, it might helps to use better crafted features that take semantics into account, instead of just shallow ones. Features that refer to the information in previous or the next sentence might be useful. Third, an pragmatic approach is to use some additional filters before applying the correction [11]. But it's more like appending an additional block which uses rule-based method in the pipeline architecture.

7 Conclusion

This project is about grammatical error correction, and the focus here is article ("a", "an", "the") errors. We use classification approach in the pipeline architecture, with each noun phrase as a candidate, and information around it as features. The experiment is 10-fold cross validations on NUCLE dataset, including Naive, Decision Tree, and MaxEnt classifiers. The highest f1 value is 0.19 by using MaxEnt classifier with duplicated-correction option, and the precision and recall are 12.8% and 34.7% respectively.

References

[1] R. Dale and A. Kilgarriff, *Helping our own: The HOO 2011 pilot shared task.* Association for Computational Linguistics, 2011.

[2] R. Dale, I. Anisimoff, and G. Narroway, *HOO 2012: A report on the preposition and determiner error correction shared task.* Association for Computational Linguistics, 2012.

[3] H. T. Ng, S. M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault, "The conll-2013 shared task on grammatical error correction," 2013.

[4] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, *The CoNLL-2014 Shared Task on Grammatical Error Correction.* 2014.

[5] D. Dahlmeier, H. T. Ng, and S. M. Wu, *Building a large annotated corpus of learner english: The nus corpus of learner english.* 2013.

[6] R. H. Susanto, "Systems combination for grammatical error correction," 2015.

[7] S. Bird, E. Klein, and E. . . Loper, *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[8] D. Klein and C. D. Manning, *Accurate unlexicalized parsing.* Association for Computational Linguistics, 2003.

[9] C. Sun, X. Jin, L. Lin, Y. Zhao, and X. Wang, *Convolutional Neural Networks for Correcting English Article Errors.* Springer, 2015.

[10] M. Garimella, "Detecting article errors in english learner essays with recurrent neural networks," 2016.

[11] D. Dahlmeier, H. T. Ng, and E. J. F. Ng, *NUS at the HOO 2012 Shared Task.* Association for Computational Linguistics, 2012.