

Grammatical Error Correction based on Domain Adaptation

Yitao Liu

Zhejiang Guangshan Vocational and Technical
University of Construction, Zhejiang, China
jackfly434@foxmail.com

Mark Dras

Macquarie University
Sydney, Australia
mark.dras@mq.edu.au

Abstract

A common issue for grammatical error correction (GEC) is how to combine the native corpus and the corpus from English as Second Language (ESL) learners together to train the GEC model. For example, though can be trained by the native corpus only, a GEC classifier performed better when trained by the ESL corpus. However, due to the small quantity of the ESL corpus, the native corpus needs to be utilized as well to solve the data-limitation problem. Unlike some previous works which combined them in specific ways or using specific classifiers, we consider this as a domain adaptation problem and provide a common method. It is based on FRUSTRATINGLY EASY DOMAIN ADAPTATION (Daumé III, 2007), which augments the feature vectors directly to improve the classifier. We examine this method for correcting article errors along with a number of baseline systems, and prove that it performs effectively when using appropriate classifiers.

1 Introduction

Grammatical Error Correction (GEC) aims to detect grammatical errors and misuse, and provide their corrections automatically. Generally speaking, two kinds of corpora can be used for GEC model training, the native-speaker English corpus, which is presumed to be all correct and easy-to-acquired (i.e. large in quantity), and the annotated corpus from English as Second Language (ESL) learners, which is closer to the real situation, while limited in quantity since it needs correction and annotation by professionals. Both of them have their own strengths and weaknesses when used for GEC model training.

Take the classification paradigm (Dale and Kilgarriff, 2011; Dale et al., 2012; Ng et al., 2013, 2014) as an example: in this setup, a GEC system detects where potential errors occur that are of a specific error type (e.g. article or preposition), and uses a particular classifier trained for that error type to deal with those potential errors. One of the attractions of GEC classifiers is that they can use the native corpus to be the whole training data. In this situation, the correction approach is similar to “fill in the blank” based on the context of the blank where the context is error-free (Rozovskaya et al., 2017). However, when correcting ESL texts, since the errors can be multiple and nested, it is hard to ensure the context is error-free, which decreases the performance of the classifier trained by native corpus only. In contrast, an ESL corpus is closer to the real situation, making the GEC classifier trained by that able to learn the specific error patterns made by ESL writers and, therefore, improve the performance of the classifier (Gamon, 2010; Han et al., 2010; Dahlmeier and Ng, 2011). The weakness of this method is the quantity of usable ESL data is much smaller, making it hard to use an ESL corpus only to train a reliable classifier. Thus, how to use those two kinds of corpora together is a research problem worth solving.

Some previous works (Rozovskaya and Roth, 2010b,a, 2011; Rozovskaya et al., 2017, for example) have combined native corpora and ESL corpora to train GEC classifiers in specific ways or using specific classifiers. In this research, we propose a common method. Our proposal is to recognize that this is, in fact, an instance of the more general problem of *domain adaptation*. The source domain data for GEC classifier training is the native corpus, and the target one the ESL corpus. We expand on this idea, and propose a solution using the method of FRUSTRATINGLY EASY DOMAIN ADAPTATION (FEDA) (Daumé III, 2007).

For the structure of this paper, in Section 2 we describe several prior data augmentation methods to combine native corpora and ESL corpora, and point out how these are specific instances of the more

general issue of domain adaptation. Section 3 gives detailed information about how we apply domain adaptation (the FEDA method specifically) to combine source and target domain data together to train the classifier. Section 4 describes the detailed experimental information. Section 5 shows experiment results and analysis. Finally, Section 6 summarizes this work and provides our future works.

2 Related Work

2.1 Artificial Errors Method

Rozovskaya and Roth (2010a) proposed the ARTIFICIAL ERRORS method to generate artificial errors from native corpus to train the classifier. The core idea for this method is to use artificial erroneous sentences to train discriminative classifiers whose errors are generated at a rate that reflects the errors made by ESL learners in order to simulate error patterns in the ESL corpus. The method can be illustrated as follows.

First of all, a confusion matrix C_{AE} is built from the ESL corpus for a specific error type that a classifier needs to correct in order to express the error pattern of the ESL learners. Each cell of C_{AE} represents $\text{Prob}(\text{source} = s \mid \text{label} = l)$, given the source sentences and their corresponding labels. Following that, a large native language corpus can be used to generate (synthetic) erroneous sentences to be the training data following the error distribution shown in C_{AE} . For example, in Rozovskaya et al. (2017), label a corresponds to source ϕ (non-article) and *the* in 9.9% and 1.6% of the cases respectively. Therefore, in the native corpus, 9.9% of the cases which used to be a are changed to ϕ , and 1.6% of them are changed to *the*. Labels of all modified cases remain as the previous labels without change. After this, the articles will be “incorrect” with the same frequency and distribution as those made by ESL writers.

Following the steps, the artificial errors method generates article errors with the same frequency and distribution of naturally occurring errors. Rozovskaya and Roth (2010a) trained several classifiers to correct article errors with ESL corpora from different first language backgrounds, and the best of them got a 16.05% error reduction (denoting the percentage reduction in the number of errors when compared to the number of errors in the ESL data), while the classifier trained on the corpus without errors got a 5.92% error reduction. However, due to the error sparsity of the ESL corpus itself, this method suffers from the low recall problem (Rozovskaya and Roth, 2010a; Rozovskaya et al., 2012), which harms its performance. Therefore, this method is not chosen to be one of our baseline systems.

2.2 Error Inflation Method

To address the error sparsity problem, Rozovskaya et al. (2012) proposed the ERROR INFLATION method. The key idea is to generate additional error cases by decreasing the proportion of correct cases and distributing the extra probability to other confusion candidates in an appropriate proportion that follows the relative distribution of each confusion candidate except the correct one in the ESL corpus. This method inflates the error rate in the artificial erroneous corpus and keeps the relative proportion of each incorrect confusion candidate the same as before, which can help the classifier to get a better recall performance.

The error inflation method firstly calculates the confusion matrix C_{AE} , which is the same as the artificial errors one. Then an inflation constant C is set, taking a value from 0 to 1, where 0 means no correct case is in the corpus and 1 means the error cases are not inflated. The correct cases in the ESL corpus are incorrect controlled by C and C_{AE} , where C is to keep the inflated proportion, and C_{AE} is to decide which incorrect candidate is used to replace the correct case in order to keep the relative distribution of the incorrect cases same as the original ESL corpus one.

Rozovskaya et al. (2012) proposed this error inflation method to build their determiner-correcting classifier in the HOO 2012 Shared Task (Dale et al., 2012) on Error Correction. Compared with using original data (30.75% and 28.97% respectively), using the error inflation method got a better F_1 score both for detection (34.62%) and correction (32.02%).

A shortcoming of the error inflation method, as the same of the artificial error method, is that when generating artificial error cases, the error inflation method chooses incorrect articles randomly, ignoring contexts of error cases, which may result in some error cases that rarely occur in a real situation. However, compared with the artificial errors method, the error inflation method releases the error sparsity problem and raises the performance of the GEC system. Thus, it is chosen to be one of our baseline systems.

2.3 NB Adaptation Method

Another method to use source domain data along with target domain data is to utilize some special properties of a specific classifier. Some classification algorithms like Naïve Bayes (NB) classifier contain a prior probability parameter to indicate the prior probability of each candidate. When NB is trained on native data, candidate priors represent the frequencies of the relative candidates in the native corpus, which does not provide any information on the actual distribution of errors made by ESL learners. If the prior is changed to the one which represents the frequencies of the relative candidates in the ESL corpus, it is feasible for the NB model to better fit in correcting ESL test data.

Following this idea, [Rozovskaya and Roth \(2011\)](#) proposed an NB ADAPTATION method to adapt ESL error information. Firstly, they trained an NB classifier in a traditional way by using source domain data. Then, at the evaluation or correction period, they replaced the prior probabilities of the model. Let s be a candidate appearing in the source sentence, and p the corresponding correct candidate. Then the *adapted prior* of p given s which used for replacing the original one is as:

$$\text{prior}(p, s) = \frac{C(s, p)}{C(s)} \quad (1)$$

where $C(s)$ is the count of s appearing in the ESL corpus, and $C(s, p)$ is the count of the occurrences where p is the correct candidate while s appears in the source sentence.

[Rozovskaya et al. \(2017\)](#) tested this method for correcting articles, prepositions, and verb agreement, and the results demonstrated that, by using the NB-adaptation method, the performance of the classifier improved clearly. For classifiers using native corpus only, the F_1 scores are 33.64%, 28.16%, and 37.00% respectively for each error type, and for NB-adopted classifiers, 35.17%, 30.69%, and 39.20% respectively. The main problem with this method is the model architecture is limited. If the classifier contains no prior parameter, it cannot apply this method. Consequently, this method is not a member of our baselines.

2.4 Language Modeling Method

Compared to the NB adaptation method, which uses target domain data to generate a special feature deployed in a source domain data trained model, the Language Modeling method uses the opposite idea. This method is based on using the output of a language model (LM) trained on the source domain data as a feature to add into the target domain classifier. The LM is used to judge whether a sentence is likely to be common, that is to say, to be correct. Given a sequence $S = w_1, \dots, w_m$ with length m , a probability distribution $p(w_1, \dots, w_m)$ over sequences of words is assigned to quantify how frequent this sequence is. In practice, these probabilities are smoothed and replaced with their corresponding log values and turned to the perplexity (ppl) The ppl PP of a discrete probability distribution p is defined as

$$PP(p) = 2^{H(p)} = 2^{-\sum_S p(S) \log_2 p(S)} \quad (2)$$

where $H(p)$ is the entropy (in bits) of the distribution.

More specifically, the source domain data is used to train an LM to give a ppl to show how the candidate sentence is likely to be a common (correct) sentence. Then the ppl is added as a feature into the feature set of the discriminative classifier, and train the classifier using target domain data. The additional feature f_{LM} is generalized as:

$$f_{LM} = \left(CW_{\min(ppl)}, \frac{\min(ppl)}{\sum ppl} \right) \quad (3)$$

where ppl means the set of ppl values of all candidate sentences, which is a pair of values, the first one, $CW_{\min(ppl)}$, means the candidate words which occurs minimum ppl for all candidate sentences is the minimum value of all candidate sentences, and the second one, $\frac{\min(ppl)}{\sum ppl}$, aims to show the performance of the ppl of the chosen sentence normalized by the total ppl of all candidate sentences. When adding this pair of features into the feature set of the classifier, it is actually added as two individual features.

[Rozovskaya and Roth \(2011\)](#) used the Language Modeling method as a key comparison system. Unlike the NB adaptation method, this method can be applied to a wide range of model architectures and is common to be a baseline method. Thus, it is chosen to be one of our baseline models.

3 Domain Adaptation Method

Here, we propose an approach to combining a native English corpus and an ESL corpus which is based on the insight that the approaches described in Section 2 are specific instances of domain adaptation. The aim of domain adaptation is to extract models from one or more domains (the source domain(s)) to apply to another domain (the target domain) in order to improve the performance of the system in the target domain. In general, the source domain data is much easier to get, which means it is large in quantity, while the target domain data has a relatively higher cost, which means it is relatively small in quantity. In our application, the source domain data is the native English corpus, and the target domain data is the annotated ESL corpus.

The method utilized in this research for domain adaptation is called FRUSTRATINGLY EASY DOMAIN ADAPTATION (FEDA) (Daumé III, 2007). This method improves a machine learner’s performance in a target domain by augmenting the feature set. Firstly, the FEDA method extracts features from the source domain data and the target domain data as normal. Secondly, it augments them in three parts: a general version, a source-specific version, and a target-specific version. The general and the source-specific versions of the augmented source domain vectors are the extracted feature vectors, and the target-specific version is the zero vector. The general and the target-specific versions of the augmented target domain vectors are the extracted feature vectors, and the source-specific version is the zero vector. Finally, the model is trained by all augmented source and target domain vectors together.

More formally, using the notations of Daumé III (2007), suppose that \mathcal{X} and \mathcal{Y} are the input space and output space respectively. D^s and D^t are the source domain data set and the target domain data set following their similar but distinct distributions respectively. D^s is a collection of N examples and D^t is a collection of M examples, where typically, $N \gg M$. In most common tasks, including our work, $\mathcal{X} \in \mathbb{R}^F$ for some $F > 0$. And the goal for domain adaptation is to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ with low expected loss when dealing with the target domain data.

The FEDA method augments the input space to be three times larger than the original one, making $\bar{\mathcal{X}} \in \mathbb{R}^{3F}$. The representation vectors of the source and target data are augmented by the FEDA method following mappings $\Phi^s, \Phi^t : \mathcal{X} \rightarrow \bar{\mathcal{X}}$ as given in Equations 4:

$$\begin{aligned}\Phi^s(\mathbf{x}) &= \langle \mathbf{x}, \mathbf{x}, \mathbf{0} \rangle \\ \Phi^t(\mathbf{x}) &= \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle\end{aligned}\tag{4}$$

where $\mathbf{0} = \langle 0, 0, \dots, 0 \rangle \in \mathbb{R}^F$ is the zero vector.

Here is a simple example to describe the application of this method for the GEC task. Sentences 1a and 1b are two instances from source and target domain data respectively. We mark out the first two articles in these two sentences.

- (1) a. I decided to buy **a** CD-ROM driver for **the** old computer in the computing laboratory.
- b. They call them **a*/ ϕ** “mothers of **the** forest ,” the tall , distinctively white-barked trees with round leaves like little sails that tremble in the slightest breeze .

The source domain sentence (1a) contains no errors, while there is one article error in the target domain sentence (1b): the first article *a* in the original sentence is incorrect, which should be corrected to ϕ as shown in the instance. It means the second article *the* should be kept without any modification in both source and target domain data. On the other hand, the first article *a* in the source domain data should be generalized to *a* since it is a correct choice, while in the target domain data, the first article *a* should be generalized to ϕ , which is different from the output of the source domain data.

Suppose there is a GEC model aiming to correct the first two articles, which only contains a very simple feature set: just take the first two articles, A_1 for the first article and A_2 for the second article in a sentence, and see whether the article in each position is *a* and *the* respectively. It can be formalized as the input space $\mathcal{X} = \langle x_0, x_1 \rangle \in \{0, 1\}^2$, where element x_0 indicates if A_1 is *a* and element x_1 indicates if A_2 is *the*. In this situation, if A_1 is *a* in the source domain data, then the correction remains *a*, if A_1 is *a* in the target domain data, then the correction is ϕ , and if A_2 is *the* no matter in source or the target domain data, then the correction remains *the*.

In the original feature set, both **1a** and **1b** will be extracted and transformed to the representation vector as $\langle 1, 1 \rangle$ without distinction, while their intended outputs are different (*a* and *the* for the source domain data, and ϕ and *the* for the target domain data). The GEC model cannot make the correct decision based on this ambiguous representation only. While after the augmentation using FEDA method, the representation vectors of **1a** and **1b** become $\langle 1, 1, 1, 1, 0, 0 \rangle$ and $\langle 1, 1, 0, 0, 1, 1 \rangle$ respectively. In this situation, if the GEC model wants to correctly learn the goal function $h : \mathcal{X} \rightarrow \mathcal{Y}$ (i.e. to change the first article *a* in target domain data to ϕ , keep the first article *a* in source domain data, and keep the second article *the* regardless the domain of the data), the weight vector for the second article *the* can be set to be something like $\langle 0, 1, 0, 0, 0, 0 \rangle$. Such a high weight on that position indicates that the second article *the* most likely needs no change, regardless of the domain. On the other hand, the weight vector for the first article *a* which needs to stay the same might be something like $\langle 0, 0, 1, 0, 0, 0 \rangle$, indicating that this *a* needs correction only in the source domain. Similarly, the weight vector for the first article *a* which needs to change to ϕ might be something like $\langle 0, 0, 0, 0, 1, 0 \rangle$, indicating that this *a* needs correction only in the target domain.

4 Experiments

4.1 Experimental Setup

4.1.1 Data Sets

Source domain data We choose the Gigaword English Corpus¹ (Graff et al., 2003) as our native English corpus. The Gigaword English Corpus is a comprehensive archive of newswire text data that has been acquired over several years by the Linguistic Data Consortium at the University of Pennsylvania. All documents in the corpus are categorized into four distinct types: *story*, *multi*, *advis*, and *other*. Among them, story type is the most typical newswire item which contains a coherent report on a particular topic or event, consisting of paragraphs and full sentences. The text content consists of pure ASCII tokens separated by white space without additional tags or special characters. All of the above makes documents in story type particularly appropriate for source domain data, which is about 5 GB in size.

Target domain data The main freely available ESL corpora include the NUCLE corpus (Dahlmeier et al., 2013) and the FCE corpus (Yannakoudakis et al., 2011). We choose the FCE corpus as the ESL learners' corpus, which here constitutes the target domain data. The FCE corpus is a subset of the CLC, a large non-public collection of texts produced by ESL learners taking Cambridge Assessment's English as a Second or Other Language (ESOL) examinations from all around the world. The released corpus consists of 1244 scripts collected from the FCE exams between 2000 and 2001, which assessed the proficiency of ESL learners for an upper-intermediate English language level (Rei and Yannakoudakis, 2016). Each exam script contains two essays in response to a prompt for a learner to write a letter, a report, an article, a composition, or a short story, about 120 to 180 words each.

4.1.2 Error Type for Processing

Based on the limitations of GEC systems using a classification framework, i.e. detecting one error type, then correcting all detected errors by using a specific classifier, it is more suitable for the GEC system to correct shallow and easily classified error types. Here, we work with the article errors because, firstly, article errors are the most frequent errors made by ESL learners. According to the CoNLL-2013 Shared Task (Ng et al., 2013) report, more than 40% of the errors among the five main error types in the test data are ArtOrDet errors. Secondly, the confusion set for article errors is quite limited (only containing ϕ , *a*, *an*, and *the*). Thus, for processing only one error type, article error is the most appropriate one.

4.1.3 Test Data and Evaluation

CoNLL-2013 (Ng et al., 2013) and CoNLL-2014 (Ng et al., 2014) shared tasks are the two main shared tasks for GEC research. In this research, the test data is from CoNLL-2013 Shared Task. Compared with CoNLL-2014 Shared Task, CoNLL-2013 Shared Task offers not only the corrections for all errors in the test data, but also the gold standard for five main error types, including ArtOrDet, Prep, Nn, Vform, and SVA, which is more suitable for our research: only correct article errors.

¹<https://catalog.ldc.upenn.edu/LDC2003T05>

The evaluation metrics used in this research are the same as those used in the CoNLL-2013 Shared Task, which contains precision, recall, and F_1 score (the choice of F score is different from the CoNLL-2014 Shared Task). The MaxMatch (M2) scorer² (Dahlmeier and Ng, 2012) was used as the official scorer.

4.1.4 Classifiers

We use Averaged Perceptron (AP), Naïve Bayes (NB), and Stochastic Gradient Descent (SGD) as our classifier models. AP and NB are chosen since they were used in Rozovskaya et al. (2017) (see Section 2), and SGD is added as an additional one for generality. What is more, all of them can use a partial fit strategy to fit big data by using an out-of-core approach, which would be very helpful for training classifiers on the large source domain data.

4.1.5 Feature Set

The feature set we choose is described in Table 1. It is simple but effective, which includes 3-window word unigrams to 3-grams features containing the detected word and their corresponding PoS features.

Feature Type	Description
Word n-grams	$w3B, w2B, w1B, w, w1A, w2A, w3A,$ $w3Bw2B, w2Bw1B, w1Bw, ww1A, w1Aw2A, w2Aw3A,$ $w3Bw2Bw1B, w2Bw1Bw, w1Bww1A, ww1Aw2A, w1Aw2Aw3A$
PoS n-grams	$p3B, p2B, p1B, p, p1A, p2A, p3A,$ $p3Bp2B, p2Bp1B, p1Bp, pp1A, p1Ap2A, p2Ap3A,$ $p3Bp2Bp1B, p2Bp1Bp, p1Bpp1A, pp1Ap2A, p1Ap2Ap3A$

Table 1: Feature set of all baseline classifiers and domain adaptation classifiers, where w means the detected word, p means the PoS of w , B and A denote the word or the PoS before and after w respectively; and the number before B and A denotes the distance of the word or the PoS from w .

4.1.6 Training and correcting steps

For the training steps, the GEC system firstly generalizes tokenization, PoS tagging, and chunking of the text by using OpenNLP³ (Apache Software Foundation, 2014), a machine learning-based toolkit for the generation of syntactic and semantic analysis of natural language texts. Then it detects the article errors by the following method: for target domain data, the system searches all article errors annotated by the FCE corpus, and records annotated articles as labels. For source domain data, following the work of HAN et al. (2006), the system searches all points where there was probably an article. More specifically, all points are searched where there is an article (*a, an, the*), and at the beginning of a noun phrase, which is a potential ϕ , and records the original articles at that point as labels, including ϕ . For each point, the system extracts features described in Table 1, and tokens are extracted only when it shows more than once. After extraction, there is a feature selection that performs a χ^2 test to the samples to retrieve the top 80% effective features to train classifiers. After all this preprocessing, the feature vectors are used to train AP, NB, and SGD classifiers: making the prediction, and then comparing the prediction with the gold standard label given by the training data to modify the parameters in classifiers. All feature selection and classifier training are realized by scikit-learn⁴ (Pedregosa et al., 2011), a simple and efficient tool for predictive data analysis, powerful for classification, Regression, Clustering, etc., and corresponding data cleaning and preprocessing.

For the correcting step, methods for generalizing text features, processing error detection, and extracting feature vectors for the classifier are the same as in the training step. For each point, the GEC system extracts features for classifiers to make the prediction, and then inserts or replaces the previous article by the prediction, except the article position that is rule-based to have no article, such as the position before a proportion or a number.

²<http://www.comp.nus.edu.sg/~nlp/software.html>

³<https://opennlp.apache.org/>

⁴<https://scikit-learn.org/stable/index.html>

4.2 Experimental Design

4.2.1 Baselines

To compare the performance of our method, we implement a number of baselines.

Target domain data baseline, which uses the target domain data only as training data, ignoring the source domain one. The system firstly extracts pure texts from the FCE corpus as the target domain data, carries out Tokenization, PoS Tagging, and Chunking of the text, searches all errors pointed by the FCE corpus, and extracts features to train AP, NB, and SGD classifiers as described in Section 4.1.6.

Source domain data baseline, which uses the source domain data only as training data, ignoring the target domain one. Due to the large size of the source domain data, we use the out-of-core technique to train classifiers. The system carries out Tokenization, PoS Tagging, and Chunking of the text, then transforms contexts to vectors by signed 32-bit version of Murmurhash3 hashing strategy, and trains AP, NB, and SGD classifiers using the partial fit technique. The hashing process is realized by scikit-learn.

All data baseline, which trains the GEC model by using both of the source and the target domain data with a random sequence. It is a simple combination without any merge technique. We chose the same hashing strategy as the Source domain data baseline used. Other training steps are the same as above.

Language Modeling baseline, which trains the GEC model by using both the source and the target domain data following the Language Modeling method (see Section 2.4). The system firstly searches all errors indicated in the FCE corpus, then deletes the previous article, inserts four kinds of articles (ϕ , a, an, the) to generalize four candidate sentences. After that, the system calculates the ppl scores for the four candidate sentences by SRILM⁵, and the feature is generalized as Equation 3, and added into the feature set along with other features mentioned above to train AP, NB, and SGD classifiers.

Error Inflation baseline, which trains the GEC model by using both the source and the target domain data following the Error Inflation method (see Section 2.2). To compare with the Target domain data baseline, the data set we randomly chose is a part of Gigawords corpus and keeps the number of article cases similar to the FCE corpus, which is around 5k. The confusion matrix C_{AE} for article errors in FCE corpus is shown in Table 2. We test three different inflation constants for reducing correct examples by 0.9, 0.7, and 0.5. For all classifiers (AP, NB, SGD), we implemented two variants cases, where they do or do not use feature selection. The feature selection group chose the top 80% effective vectors to optimize each classifier.

Label \ Source	ϕ	a	an	the
ϕ	0.990	0.009	0.007	0.013
a	0.034	0.853	-	0.018
an	0.035	-	0.874	0.023
the	0.046	0.009	0.001	0.916

Table 2: Confusion Matrix for article errors in FCE corpus. The left column shows the correct article, and each element in a row shows the probability of article the author may choose $Prob(source|label)$.

4.2.2 Domain Adaptation Method

In order to unify the input space for both source and the target domain data, we chose the same hashing strategy to transform the source and the target domain data into vectors. Specific methods follow the methods described above for the source domain data baseline. After that, the vectors are augmented to train the augmented vector classifiers, for each of NB, AP, and SGD. As mentioned in Section 3, the source domain data vectors and the target domain data vectors are augmented following the mapping $\Phi^s(\mathbf{x}) = \langle \mathbf{x}, \mathbf{x}, \mathbf{0} \rangle$ and $\Phi^t(\mathbf{x}) = \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle$ respectively.

When generating predictions for the test data, we transform texts into vectors by the same hashing strategy, and augmented them following the mapping $\Phi^t(\mathbf{x}) = \langle \mathbf{x}, \mathbf{0}, \mathbf{x} \rangle$, since the test data should be treated as the target domain data.

⁵<https://www.sri.com/case-studies/srilm/>

5 Results and Analysis

5.1 Baselines

Source / Target / All Domain The results for Source domain data, Target domain data, and All data baseline systems are shown in Table 3. We make three brief observations. Firstly, the NB classifier got the best result in the Target domain data method, and the SGC performed the best in remaining two methods. Secondly, for the AP and NB classifiers, due to the dramatically fell of recall, the performance dropped substantially when source domain data was involved in the training data (the Source domain data method and the All data method). Thirdly, for the SGD classifier, results for those three baseline systems remained relatively stable, while the inclusion of the target domain data negatively influenced the SGD classifier.

Baselines	Model	Precision	Recall	F ₁ score
Target domain data	AP	0.102	0.282	0.150
	NB	0.110	0.294	0.160
	SGD	0.098	0.281	0.146
Source domain data	AP	0.084	0.017	0.029
	NB	0.093	0.042	0.058
	SGD	0.168	0.183	0.175
All data	AP	0.080	0.019	0.031
	NB	0.095	0.039	0.055
	SGD	0.159	0.169	0.164

Table 3: Results for Source domain data, Target domain data, and All data baseline methods. Noticing that the best result for each baseline method is highlighted in bold.

Language Modeling The results for Language Modeling baseline systems are shown in Table 4. As can be seen, for AP and NB classifiers, the performances were similar to the Target domain data baseline, and the SGD classifier clearly performed worse, with both the precision and the recall dropping a lot. The results showed that the Language Modeling method led to a negative influence for GEC classifiers.

Model	Precision	Recall	F ₁ score
AP	0.101	0.286	0.150
NB	0.106	0.291	0.155
SGD	0.065	0.184	0.096

Table 4: Results for the Language Modeling baseline method. The best result is highlighted in bold.

Error Inflation The results for Error Inflation baseline systems which contained three different inflation constants and with / without feature selection are shown in Table 5. Three points need to be mentioned. Firstly, when the inflation constant is 0.7, AP classifiers performed the best regardless of other parameters, while for NB and SGD classifiers, the most appropriate inflation constant is 0.5. Secondly, AP and SGD classifiers with feature selection performed superior to those without in all three metrics, while NB classifiers were the opposite. Thirdly, the best performed classifier of all ones using the Error Inflation method was the NB with 0.5 inflation constant and without feature selection (0.177 in F₁ score).

Overall Comparison Here we make a overall comparison for all baselines. Firstly, the Error Inflation method got the best performance in all three types of classifiers. Despite the results of the Error Inflation method were chosen for the best one from different hyper-parameters, if we choose fixed hyper-parameters, the situation remained the same: 0.5 inflation constant with feature selection (0.5fs) for the Error Inflation method still got similar results (0.161 for AP, 0.173 for NB, and 0.176 for SGD) compared with the all-best results (0.168 for AP, 0.177 for NB, and 0.176 for SGD), and were all superior to corresponding classifiers using other baseline methods.

Model	Precision	Recall	F ₁ score	Model	Precision	Recall	F ₁ score
AP0.9	0.115	0.067	0.085	AP0.9fs	0.125	0.080	0.098
NB0.9	0.150	0.117	0.131	NB0.9fs	0.145	0.112	0.126
SGD0.9	0.077	0.039	0.052	SGD0.9fs	0.157	0.080	0.106
AP0.7	0.159	0.131	0.144	AP0.7fs	0.188	0.152	0.168 ⁺
NB0.7	0.172	0.152	0.161	NB0.7fs	0.171	0.149	0.159
SGD0.7	0.134	0.078	0.099	SGD0.7fs	0.150	0.089	0.112
AP0.5	0.104	0.092	0.097	AP0.5fs	0.172	0.152	0.161
NB0.5	0.182	0.173	0.177 *	NB0.5fs	0.179	0.168	0.173
SGD0.5	0.192	0.149	0.168	SGD0.5fs	0.206	0.155	0.176 ⁺

Table 5: Results for the Error Inflation baseline method. The figure after the type of classifier indicates the inflation constant, and 'fs' after the inflation constant indicates the classifier had feature selection process. The best results for methods with same hyper-parameters are highlighted in bold, the best one for each type of classifier is indicated by ⁺ after the figure, and the best one of all results is indicated by *.

Secondly, the AP and NB classifiers were quite sensitive to target domain data, which can drive down their performance dramatically, while they all performed well when using the Error Inflation method. It may be because AP and NB classifiers are more sensitive to context information. When the context contained other errors except for article errors in the real situation, the word context and corresponding PoS may be more muddled, while artificial errors generated by the Error Inflation method only modified the article choice and maintained the context of erroneous articles correct, which relieved such problem.

Thirdly, compared with AP and NB, the SGD classifier performed well in terms of adapting to the real situation, i.e. the target domain data. Though worse in the Source domain method, the performance of SGD classifiers in the Target domain method and the All data method was substantially better than the related one of AP and NB classifiers. And the performance of the Target domain method for the SGD classifier was equal to the best Error Inflation method (0.175 vs. 0.176), proving that, when facing a more complicated context, the SGD classifier can be more effective than the other two.

5.2 Domain Adaptation Method

The result for the Domain Adaptation method is shown in Table 6. To compare with other baselines, we also generate a column graph of the F₁ score for all methods we conducted, which is shown in Figure 1.

Model	Precision	Recall	F ₁ score
AP	0.096	0.232	0.136
NB	0.046	0.086	0.060
SGD	0.171	0.219	0.192

Table 6: Results for the Domain Adaptation method. Noticing that the best result is highlighted in bold.

As can be seen from Table 6, SGD classifier has the best performance among the three classifiers. Compared with the Source and the Target domain data SGD classifiers from Table 3, SGD Domain Adaptation classifier improved the performance by about 10% of F₁ score for SGD Source domain data classifier (0.175), and 32% for SGD Target domain data classifier (0.146). Moreover, although the precision for the SGD Domain Adaptation classifier (0.171) is close to the SGD Source domain data classifier (0.168, a 2% improvement), the recall (21.9 and 18.3% respectively) increased significantly (a 20% improvement), which demonstrates that the Domain Adaptation method is quite effective at improving the low-recall problem. On the other hand, although the recall for SGD Domain Adaptation classifier is lower than SGD Target domain data classifier (0.281), the precision is dramatically increased (from 0.098 to 0.171, a 74% improvement), proving that the Target domain data classifier had good performance in recall, but weak in precision due to a small data quantity.

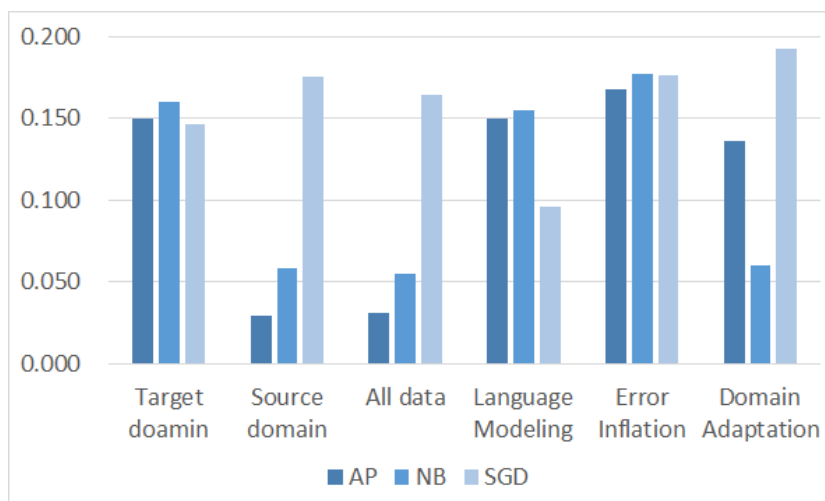


Figure 1: The column graph of the F_1 score for all baseline methods along with the Domain Adaptation method. Specially, the Error Inflation method chose the best performed case for each type of classifier, which are AP0.7fs, NB0.5, and SGD0.5fs.

For the AP and NB classifiers using the Domain Adaptation method, the performance and the improvement were inferior to the SGD Domain Adaptation classifier. Though achieved a high level of recall (0.232), the AP Domain Adaptation classifier suffered a low precision (0.096), leading to a relatively weak F_1 score (0.136), which was superior to the Target domain and All data baseline results, while inferior to other baseline results. A similar situation can be found in the NB Domain Adaptation classifier, except that the recall was uncompetitive than others (0.086), making the F_1 score (0.060) the worst in all three Domain Adaptation classifiers, and other NB baseline classifiers except Target domain and All data ones. The reason for that may be the same one as mentioned before: when the context contains other errors besides article errors in the real situation, AP and NB classifiers can be more easily influenced by other errors, while SGD classifiers are more robust for complicated context.

Considering results from all baseline systems and the Domain Adaptation method system through Figure 1, it can be found that SGD classifier with the Domain Adaptation method got the best performance among all classifiers, 0.015 better than the second-best one (NB0.5 in the Error Inflation method, 0.177 in F_1 score): the improvement over Error Inflation is similar to the improvement of Error Inflation over the Source Domain baseline. It demonstrates that, when choosing an appropriate type of classifier (SGD), the Domain Adaptation method can help the classifier to get a promising result.

6 Conclusion

In this work, we used the insight that it could be treated as a domain adaptation problem to combine source domain data and target domain data to improve the GEC classifier. Specifically, we used the FEDA method to combine a native English corpus, usually large in quantity while relatively less representative, and an ESL learners' corpus, which is more similar in error pattern in the real situation while much smaller in quantity. After comparing with a number of baselines, our method shows that, when using an SGD classifier, the performance relative to baselines is encouraging for the use of domain adaptation.

The ESL learner corpus showed its usefulness in GEC for ESL texts due to the similarity of error patterns. However, there is still room for more detailed distinctions. The target domain can be separated by the first languages of ESL learners. It is worthwhile to research the performance of classifiers trained by corpus from specific first language background as target domain data, and native corpus as source domain data together using the domain adaptation method. Another direction is to take the idea of domain adaptation to more recent approaches such as the Neural Machine Translation paradigm. In the context of deep learning, domain adaptation is more typically carried out through transfer learning. A lot of works have been done in this area, proving that it is worthy to further discover.

References

- Apache Software Foundation. OpenNLP Natural Language Processing Library, 2014.
- Daniel Dahlmeier and Hwee Tou Ng. Grammatical error correction with alternating structure optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 915–923, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
- Daniel Dahlmeier and Hwee Tou Ng. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June 2012. Association for Computational Linguistics.
- Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia, June 2013. Association for Computational Linguistics.
- Robert Dale and Adam Kilgarriff. Helping Our Own: The HOO 2011 Pilot Shared Task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 242–249, Nancy, France, September 2011. Association for Computational Linguistics.
- Robert Dale, Ilya Anisimoff, and George Narroway. HOO 2012: A Report on the Preposition and Determiner Error Correction Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada, June 2012. Association for Computational Linguistics.
- Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
- Michael Gamon. Using mostly native data to correct errors in learners’ writing: A meta-classifier approach. In *Proceedings of NAACL 2010*. Association for Computational Linguistics, June 2010.
- David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34, 2003.
- NA-RAE HAN, MARTIN CHODOROW, and CLAUDIA LEACOCK. Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129, 2006.
- Na-Rae Han, Joel Tetreault, Soo-Hwa Lee, and Jin-Young Ha. Using an error-annotated learner corpus to develop an ESL/EFL error correction system. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC’10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).
- Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
- Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

- Marek Rei and Helen Yannakoudakis. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1112. URL <https://aclanthology.org/P16-1112>.
- Alla Rozovskaya and Dan Roth. Training Paradigms for Correcting Errors in Grammar and Usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 154–162, Los Angeles, California, June 2010a. Association for Computational Linguistics.
- Alla Rozovskaya and Dan Roth. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 961–970, Cambridge, MA, October 2010b. Association for Computational Linguistics.
- Alla Rozovskaya and Dan Roth. Algorithm Selection and Model Adaptation for ESL Correction Tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 924–933, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
- Alla Rozovskaya, Mark Sammons, and Dan Roth. The UI system in the HOO 2012 shared task on error correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 272–280, Montréal, Canada, June 2012. Association for Computational Linguistics.
- Alla Rozovskaya, Dan Roth, and Mark Sammons. Adapting to learner errors with minimal supervision. *Computational Linguistics*, 43(4):723–760, December 2017.
- Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.