Addressing Class Imbalance in Grammatical Error Detection with Evaluation Metric Optimization

Anoop Kunchukuttan, Pushpak Bhattacharyya
Center for Indian Language Technology
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
{anoopk,pb}@cse.iitb.ac.in

Abstract

We address the problem of class imbalance in supervised grammatical error detection (GED) for non-native speaker text, which is the result of the low proportion of erroneous examples compared to a large number of error-free examples. Most learning algorithms maximize accuracy which is not a suitable objective for such imbalanced data. For GED, most systems address this issue by tuning hyperparameters to maximize metrics like $F_\beta$. Instead, we show that learning classifiers that directly learn model parameters by optimizing evaluation metrics like $F_1$ and $F_2$ score deliver better performance on these metrics as compared to traditional sampling and cost-sensitive learning solutions for addressing class imbalance. Optimizing these metrics is useful in recall-oriented grammar error detection scenarios. We also show that there are inherent difficulties in optimizing precision-oriented evaluation metrics like $F_{0.5}$. We establish this through a systematic evaluation on multiple datasets and different GED tasks.

1 Introduction

The task of grammatical error detection (GED) and grammatical error correction (GEC) refers to the identification and repair of grammatical errors in text generated by speakers of a language (native/non-native). Many techniques, rule-based as well as machine learning based, have been proposed for addressing this task. A common and successful method for building grammatical error detection (GED) systems is to apply supervised learning on annotated learner corpora. For instance, the problem of noun number error detection can be formulated as a task of classifying the head noun of a noun phrase has the correct or incorrect grammatical number. As opposed to rule-based methods, classification methods can easily incorporate complex and arbitrary evidence as features. Some of the best performing systems for the most frequent grammatical errors, viz. noun number, article and preposition errors, are classification systems (Ng et al., 2013).

However, a major problem in learning classifiers from an annotated learner corpus is the very low error rate (number of errors per token) in the corpus (Chodorow et al., 2012). For instance, the error rates are less than 3% for noun-number, article and preposition errors in the NUCLE annotated learner corpus (Dahlmeier et al., 2013) as shown in Table 1. Such low error rates (less than 5%) have been observed across various learner corpora (see Table 2) viz. the NUCLE corpus (Dahlmeier et al., 2013), HOO12 corpus (Dale et al., 2012) and NICT-JLE corpus (Izumi et al., 2004). Thus, learning classifiers for GEC/GED from annotated learner corpora is a case of learning a classifier from an imbalanced dataset i.e. a dataset where the class ratios are highly skewed (He and Garcia, 2009). In contrast to many imbalanced problems studied in datamining literature, GED tasks are characterized by large and sparse features spaces and very high imbalance ratios.

Since it is easy to achieve high accuracy by simply assigning all training examples to the majority class, accuracy as the optimization objective performs poorly for classification on imbal-

| Task          | Errors | Tokens   | Error Rate |
|---------------|--------|----------|------------|
| Article       | 6658   | 234,695  | 2.84%      |
| Noun Number   | 3379   | 245,026  | 1.38%      |
| Preposition*  | 1955   | 123,419  | 1.58%      |

Table 1: Error rates for GED tasks (NUCLE)
* statistics for 10 most frequent prepositions
Corpus Tokens Errors % Error Rate
NUCLE 1,161,567 46,597 3.82
HOO12 374,680 8,432 ‡ 2.25‡
NICT-JLE 169,662 14,407 8.49

Table 2: Errors statistics for learner corpora
‡ articles and prepositions errors only

anced datasets. Moreover, accuracy is not the right evaluation metric for GED. Non-native language learners require a GED system with a high precision so that they are not misguided by wrong error notifications. On the other hand, professional copy-editors and language instructors will require a high recall system that helps improve productivity and quality of service by identifying all potential errors that need review, even at the cost of flagging some spurious errors. Hence, a precision oriented optimization objective like $F_{0.5}$ and a recall oriented objective like $F_2$ would be appropriate for non-native speakers and copy-editors respectively. In this paper, we explore the hypothesis that by directly optimizing the desired evaluation metric, we can satisfy the requirements of asymmetric misclassification cost and customization of the recall/precision trade-off.

The following are the contributions of our work:

- Sampling and example-weighting methods have been traditionally applied to overcome this limitation. We systematically investigate different solutions to the class imbalance problem for three GED tasks (noun number, article and preposition) over multiple annotated learner corpora. We compare the following sampling methods over a range of sampling ratios: random undersampling, Synthetic Minority Over-sampling Technique (SMOTE) and example weighting. We analyze and present experimental results to demonstrate the limitations of these traditional methods in addressing class imbalance.

- As an alternative to sampling methods, we propose that a GED classifier be learnt by directly optimizing the evaluation metrics, typically $F_1$, $F_2$ or $F_{0.5}$. For copy-editors, $F_2$ is a suitable evaluation measure which addresses the need for high recall in GED. We use the performance measure optimization framework proposed by Joachims (2005) for optimizing these metrics. For the three GED tasks under consideration, we show that optimizing the $F_2$ metric of the incorrect class gives better performance compared to sampling methods.

- We also show that evaluation metric optimization, as well as sampling methods, are not suitable for improving precision. While evaluation metric optimization helps improve recall and obtain a reasonable precision-recall trade-off, improving precision remains a challenge.

The rest of the paper is organized as follows. Section 2 describes the limitations of sampling methods, motivates the use of evaluation metrics optimization for imbalanced data problems, and the use of $F_2$ as an evaluation metric for GED. Sections 3 and 4 discuss the related work and the learning algorithm used for directly optimizing evaluation metrics. Section 5 explains classifiers for the GED tasks under consideration, while Section 6 describes our experimental setup. We analyze results in Section 7. Section 8 summarizes our contributions and describes possible future directions.

2 Motivation

There are many GED applications where users do not consider false positive and false negative errors as equally damaging. To copy-editors, false negative errors are costlier than false positive errors. They would not mind evaluating a few false alarms, but cannot afford to miss genuine errors since that would affect the quality of service. They would prefer the GED system to err on the side of higher recall. Similarly, translators using computer-aided translation systems would prefer to have most errors pointed out. Even an automatic post-editing system for MT would prefer to have most errors identified since alternatives can then be evaluated for these potential errors in a post-editing stage.

We hypothesize that by directly optimizing the desired evaluation metric, we can satisfy the requirements of asymmetric loss and customization of the recall/precision trade-off. For a GED system designed to help copy-editors, $F_2$ would be a reasonable choice as an optimization objective since it is biased towards recall. Similarly, $F_{0.5}$ would be a reasonable choice for a GED system designed for language learners. Our choice of
\( \beta = 0.5, 2 \) follows the conventional choices mentioned in literature for evaluating precision and recall bias.

Traditionally, sampling and example-weighting have been the most common methods to handle class imbalance. **Sampling** involves either **undersampling** the correct examples or **oversampling** the incorrect examples in the training set (Van Hulse et al., 2007; Domingos, 1999). In **example-weighting**, misclassification costs are associated with the training examples, with higher misclassification costs for wrongly classifying incorrect examples (Zadrozny et al., 2003). In theory, example-weighting and sampling techniques can be shown to be equivalent (Zadrozny et al., 2003), and in the rest of the paper we use sampling to refer to both. However, sampling methods have a few limitations which direct optimization of the performance metrics does not suffer from:

- Even the choice of sampling ratio is arbitrary and its effect on the evaluation metrics is not obvious. A validation set may be used to select the appropriate ratio which maximizes the evaluation metric, at the cost of setting aside some valuable training data for tuning. On the other hand, while optimizing an evaluation metric, it naturally induces a loss function.
- A high degree of undersampling is required to negate the effect of the very low error rate in GED tasks. However, this results in a drastic reduction in the examples available for training. Random undersampling has been reported to work reasonably on some datasets with a dense feature space and a small number of features (Van Hulse et al., 2007). In contrast, the classification problems for GED results are characterized by large and sparse feature spaces leading to a loss of many features due to undersampling. Optimizing evaluation metrics directly does not incur this loss since data is not sampled prior to learning.
- Oversampling with repetition of incorrect class instances generally does not improve classification performance and results in overfitting. Informed oversampling through introduction of synthetic instances belonging to the minority class (SMOTE) by interpolation of minority class instances in the training set has shown good improvement in many applications (Chawla et al., 2002). For large feature spaces, the generation of synthetic examples can be computationally expensive since it involves a k-nearest neighbourhood search. Optimizing evaluation metrics directly does not incur this computational overhead.

### 3 Related Work

The most common methods to handle **imbalanced datasets** in GED involve undersampling the correct instances (Dahlmeier et al., 2012; Putra and Szabo, 2013; Kunchukuttan et al., 2013) or oversampling the incorrect instances (Xing et al., 2013). Rozovskaya et al. (2012) propose an error inflation method for preposition error correction, where a fraction of the correct prepositions are marked as incorrect prepositions and these new “erroneous” instances are distributed among different possible erroneous prepositions. This method is similar to the Meta-Cost (Domingos, 1999) approach of re-labelling examples in the training set according to a cost function. Some whole-sentence correction approaches (Kunchukuttan et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014; Dahlmeier and Ng, 2012) are tuned to maximize the \( F_\beta \) scores.

In all the work mentioned above, the systems maximize \( F_\beta \) by tuning only the “hyperparameters” like sampling threshold, features weights for scores of underlying components —classifier (Dahlmeier and Ng, 2012) or SMT component scores (Kunchukuttan et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014) —on a tuning dataset using approximate methods like MERT (Och, 2003), PRO (Hopkins and May, 2011) and grid search. In contrast, we optimize all the model parameters exactly and efficiently to maximize \( F_\beta \).

### 4 Optimizing Performance Measures

A loss function like the induced \( F_\beta \) loss is non-decomposable i.e., it cannot be expressed as the sum of losses over individual instances. We use the max-margin formulation proposed by Joachims (2005) for exactly optimizing such non-decomposable loss functions which can be com-
puted from the contingency table\(^1\). It is applicable only to binary classification problems.

The training loss is described in terms of a loss function \((\Delta)\) which measures the discrepancy between the expected output vector \((\bar{y})\) and observed output vector \((\bar{y}')\) on training data of size \(m\):

\[
\Delta : (y_1 \cdots y_i \cdots y_m) \times (y'_1 \cdots y'_i \cdots y'_m) \rightarrow \mathbb{R} \tag{1}
\]

For a decomposable loss functions like 0-1 loss, hinge loss, etc. we can express the loss as:

\[
\Delta(y, \bar{y}) = \sum_{i=1}^{m} \Delta'(y_i, y'_i) \tag{2}
\]

where, \(\Delta'\) is the loss function for a single instance:

\[
\Delta' : y \times y' \rightarrow \mathbb{R} \tag{3}
\]

In the max-margin framework, a decomposable loss function (hinge loss in the example below) can be minimized by solving the following objective function for support vector machines (SVM):

\[
\min \frac{1}{2} \sum_{i=1}^{n} \xi_i + C \sum_{i=1}^{m} \xi_i \tag{4}
\]

subject to the following constraints

\[
\forall \bar{y}' \in \mathcal{Y} \setminus \bar{y} : \langle \bar{x}, \delta \Psi(\bar{x}, \bar{y}') \rangle \geq \Delta(\bar{y}, \bar{y}') - \xi \tag{7}
\]

where,

\[
\delta \Psi(\bar{x}, \bar{y}') = \Psi(\bar{x}, \bar{y}') - \Psi(\bar{x}, \bar{y}) \tag{8}
\]

\[
\Psi(\bar{x}, \bar{y}') = \sum_{i=1}^{m} y'_i x_i \tag{9}
\]

Thus any label vector of the training set \((\bar{y}')\) other than the expected label vector \((\bar{y})\) would incur a loss upper bounded by:

\[
\xi = \Delta(\bar{y}, \bar{y}') + \Psi(\bar{x}, \bar{y}') - \Psi(\bar{x}, \bar{y}) \tag{10}
\]

where, \(\Delta\) is a custom loss function between the expected and observed label vectors.

The number of constraints is exponential in \(m\), since there are \(2^m - 1\) possible values of \(\bar{y}'\). However, the above optimization problem can be solved efficiently in a polynomial number of iterations using an iterative, cutting plane algorithm. The efficiency of the overall algorithm depends on the efficiency of finding the most violated constraint, i.e. finding the \(\bar{y}'\) which maximizes Equation 10. This loss-augmented inference problem can be solved in polynomial time for binary classification problems involving loss functions which can be computed from the contingency table e.g. \(F_\beta\), precision, recall loss.

### 5 Grammatical Error Detection Classifiers

The GED tasks and the classifiers under study are described in this section.

#### 5.1 Tasks

We consider the following three GED tasks: Noun Number (NN), Article (Art) and Preposition (Prep).

| Noun Number | Preposition |
|-------------|-------------|
| O: It was great to see the **excitements** on the child’s face. | O: I want to buy pen.  |
| C: It was great to see the **excitement** on the child’s face. | C: Keep the pen **in** the table.  |

**Table 3: Examples of grammatical errors**

| O: original, C: corrected |
|---------------------------|
| (O: I want to buy pen) and (C: Keep the pen on the table). |
error detection. A few examples of these errors are shown in Table 3.

Each is a binary classification task which labels an instance as grammatically correct (negative) or incorrect (positive). The training instances for each task are defined as follows. For noun number, each noun phrase is an instance. For articles, each noun phrase containing the articles {a, an, the, φ} is an instance. For preposition, we consider only preposition deletion and substitution errors. We consider only the ten most frequent prepositions as instances: {on, from, for, of, about, to, at, in, with, by} (Rozovskaya and Roth, 2010). The top 10 prepositions account for 78% of all prepositions in the NUCLE corpus and 81% of all the preposition errors. The feature sets for the GED tasks are shown in Table 4.

5.2 Directly optimizing evaluation metrics

We directly optimize $F_\beta$ using the support vector method for optimizing performance measures (SVM-Perf) proposed by Joachims (2005). We chose this method since it can exactly and efficiently optimize non-decomposable evaluation metrics which can be computed from the contingency table e.g. $F_\beta$. We train different classifiers corresponding to $\beta = 0.5, 1, 2$.  

5.3 Sampling Methods

Sampled datasets were created from the original training set using three sampling methods. In random undersampling (Van Hulse et al., 2007), the correct instances are undersampled at random and all incorrect instances are retained in the training set. In SMOTE (Chawla et al., 2002), incorrect class examples are over-sampled by generating synthetic examples for every incorrect instance using linear interpolation with one of its five nearest incorrect neighbours. In cost-sensitive learning (Zadrozny et al., 2003), misclassifying incorrect instances incurs a higher cost as compared to misclassifying correct instances.

For each sampling method, sampled datasets were created using different representative sampling ratios $(p = 0.3, 0.5, 1.0)$ which span the entire range. The sampling ratio $(p)$ refers to the ratio of incorrect to correct examples in the sampled dataset. For cost-sensitive learning, the corresponding misclassification cost ratio $(J)$ can be computed as $J = pR$, where $R$ is the ratio of correct to incorrect instances in training set. An SVM is trained with hinge loss on each of these sampled datasets.

6 Experimental Setup

We tested our GED systems on three annotated learner corpora: NUCLE (Dahlmeier et al., 2013), HOO11 (Dale and Kilgarriff, 2011) and HOO12 (Dale et al., 2012) shared task corpora. For the HOO12 dataset, noun number error detection was not done since the dataset did not have these annotations.

For hinge loss, the classifiers were trained using the SVMLight package (Joachims, b). For other loss functions, classifiers were trained using the SVM-Perf package (Joachims, a) with extensions to optimize recall, precision and $F_\beta$. We used a linear kernel for all our experiments. The evaluation was done using Precision, Recall, $F_{0.5}$, $F_1$ and $F_2$ metrics. The average scores over a 5-fold cross-validation are reported.

7 Results and Discussion

7.1 Limitations of sampling methods

Figure 1 shows the $F_2$ scores of sampling methods for different GED tasks on the NUCLE dataset as a function of the sampling ratio. We can see
that the scores vary sharply with sampling ratio, making the choice of the right sampling ratio extremely important. Undersampling techniques show a sharp increase in recall with increasing sample ratios (up to 20%), but a sharp drop in precision with increase in sampling ratio (halving the precision in some cases). Thus, the methodology does not offer much in terms of achieving a good precision-recall trade-off. The drastic variation indicates that the bias introduced by the sampling ratio is driving the classifier’s decision and the training data’s role is made irrelevant due to sampling loss.

In general, we can observe a trend that undersampling and cost-sensitive techniques perform best at low sampling ratios (p=0.3), whereas SMOTE performs best at higher sampling ratios (p=1.0). This is a consequence of the high imbalance ratio for GED tasks. However, there is no sampling method that is uniformly best across tasks and evaluation metrics. Empirical cross-validation is the only way to determine the best sampling method and configuration for a particular task.

### 7.2 SVM-Perf vs. sampling methods

Table 5 shows $F_1$, $F_2$ and $F_{0.5}$ evaluation for various SVM-Perf classifiers (optimized for $F_2=0.5,1,2$) and compares them with the classic hinge loss classifier and the best undersampling, oversampling and cost-sensitive learning methods on three datasets.

For the $F_2$ evaluation metric, optimizing $F_2$ is clearly better than the sampling techniques in all

| Dataset | Method | Noun Number $F_{0.5}$ | Noun Number $F_1$ | Noun Number $F_2$ | Article $F_{0.5}$ | Article $F_1$ | Article $F_2$ | Preposition $F_{0.5}$ | Preposition $F_1$ | Preposition $F_2$ |
|---------|--------|------------------------|-------------------|-------------------|------------------|----------------|------------------|------------------|------------------|------------------|
| NUCLE   | Hinge Loss | 7.80 | 3.42 | 2.19 | 6.92 | 3.07 | 1.98 | 15.37 | 7.85 | 5.27 |
| NUCLE   | Random Undersampling (p=0.3) | 11.01 | 15.62 | 26.89 | 14.57 | 20.01 | **31.94** | 6.75 | 9.53 | 16.23 |
| NUCLE   | Cost-Sensitive (p=0.3) | 10.46 | 13.43 | 18.79 | **18.58** | 22.15 | 29.33 | 13.53 | 8.14 | 5.82 |
| NUCLE   | SMOTE (p=1.0) | **18.91** | 14.74 | 12.27 | 10.50 | 5.78 | 4.00 | 10.82 | 5.28 | 3.52 |
| NUCLE   | SVM-Perf $F_{0.5}$ | 14.93 | 19.17 | 26.82 | 17.74 | 22.54 | 30.17 | 6.39 | 9.41 | 17.84 |
| NUCLE   | SVM-Perf $F_1$ | 15.72 | **19.92** | 27.24 | 17.95 | **22.65** | 30.72 | 6.60 | **9.70** | **18.28** |
| NUCLE   | SVM-Perf $F_2$ | 13.72 | 18.43 | **28.08** | 18.02 | 22.65 | 30.51 | 6.55 | 9.63 | 18.15 |

| HOO-11  | Hinge Loss | 18.48 | 14.15 | 11.64 | **35.55** | 29.87 | 25.85 | **28.90** | 16.92 | 12.22 |
| HOO-11  | Random Undersampling (p=0.3) | 3.55 | 5.36 | 10.98 | 14.64 | 19.54 | 29.42 | 9.90 | 11.96 | 15.20 |
| HOO-11  | Cost-Sensitive (p=0.3) | **18.48** | 14.15 | 11.64 | 33.88 | 29.49 | 26.23 | **28.90** | 16.92 | 12.22 |
| HOO-11  | SMOTE (p=1.0) | 18.48 | 14.15 | 11.64 | 34.28 | 29.64 | 26.19 | **28.90** | 16.92 | 12.22 |
| HOO-11  | SVM-Perf $F_{0.5}$ | 4.42 | 6.45 | 11.98 | 25.86 | 27.89 | 30.44 | 26.67 | **17.69** | 13.32 |
| HOO-11  | SVM-Perf $F_1$ | 4.42 | 6.45 | 11.98 | 24.06 | 28.12 | 33.88 | 24.84 | 17.19 | 13.21 |
| HOO-11  | SVM-Perf $F_2$ | 4.70 | 6.96 | **13.42** | 18.58 | 24.01 | **34.05** | 12.69 | 14.39 | **16.81** |

| HOO-12  | Hinge Loss | 7.66 | 3.74 | 2.47 | 17.63 | 10.14 | 7.12 |
| HOO-12  | Random Undersampling (p=0.3) | 10.11 | 13.75 | 21.50 | 12.26 | 15.91 | 22.66 |
| HOO-12  | Cost-Sensitive (p=0.3) | **14.66** | **16.79** | 19.65 | 17.58 | 11.47 | 8.51 |
| HOO-12  | SMOTE (p=1.0) | 7.28 | 3.90 | 2.66 | 12.74 | 6.83 | 4.68 |
| HOO-12  | SVM-Perf $F_{0.5}$ | 11.37 | 15.05 | **22.29** | 12.74 | 17.01 | **25.42** |
| HOO-12  | SVM-Perf $F_1$ | 11.69 | 15.22 | 21.83 | 10.84 | 15.11 | 24.91 |
| HOO-12  | SVM-Perf $F_2$ | 11.94 | 15.48 | 22.02 | 9.48 | 13.80 | 25.32 |

Table 5: Comparison of SVM-Perf with best sampling methods (scores in %)

Figure 1: Comparison of sampling methods on NUCLE dataset (scores in %)
but one out of 8 cases (NUCLE, Art). Even in this case, the performance of \( F_\beta \) optimization is comparable to the best sampling method (96% of the undersampling score). Improvements up to 15% in \( F_2 \) score over the best sampling techniques have been observed. For instance, on the HOO11 dataset, the \( F_2 \) scores improve by about 15%, 15% and 10% over the best sampling methods for the NN, Art and Prep tasks respectively. Using \( F_2 \) optimization, the precision is generally higher than that of the best performing sampling technique. Only in 3 cases, \( F_2 \) optimization gives slightly lower \( F_2 \) scores than \( F_1 \) or \( F_{0.5} \) optimizations (≈ 1% lesser). A GED system designed to achieve high recall can thus benefit from optimizing the \( F_2 \) metric.

\( F_1 \) optimization gives the best \( F_1 \) score on the NUCLE dataset for all tasks, and on some tasks on other datasets. For instance, the \( F_1 \) scores improve by 27.52%, 2.26% and 1.78% over the best sampling methods for the NN, Art and Prep tasks respectively on the NUCLE dataset. Only on the NN task for the HOO-11 dataset, the sampling methods vastly outperform \( F_1 \) optimization. On the remaining tasks, the performance is comparable to the best performing sampling method. In most cases, we can conclude that \( F_1 \) optimization would yield a good \( F_1 \) score.

Finally, none of the \( F_\beta \) optimizers perform well for \( F_{0.5} \) as the evaluation metric. The precision/recall analysis in the next section explains this issue.

### 7.3 Precision, Recall and Accuracy Analysis

Table 6 shows the accuracy, precision and recall for the classic hinge loss classifier and the best undersampling, oversampling, cost-sensitive and SVM-Perf classifiers on the NUCLE dataset. Other datasets also show similar trends.

Undersampling methods achieve higher recall (more than 70% for all tasks with \( p = 1.0 \)), which can be attributed to the strong inductive bias that alters the class prior in favour of the minority class. But there is a substantial reduction in precision and accuracy. The classic SVM with no sampling achieves the highest precision for all tasks (more than 40% for all tasks), while SMOTE also shows higher precision (between 25-32%). But the recall is as low as 5%. The SVM-Perf classifier maintains a comparatively high precision (≈ 15%, except preposition GED task) as well recall (between 35-45%), while the accuracy drop compared to the classic hinge is comparatively less.

We also investigated if SVM-Perf is effective in obtaining high precision or recall by directly optimizing precision and recall respectively (see Table 7 for results). Optimizing recall significantly increases the recall over \( F_\beta \) optimization, with recall of more than 50% achieved on all tasks. However, optimizing precision does not show much improvement over the \( F_\beta \) optimization and is clearly far worse than the precision obtained with the baseline SVM. Precision loss can be represented as \( TP/(FP+TP) \). We can see that \( FP = 0 \) can easily be achieved by assigning all examples to the correct class, since \( FN \) does not affect the precision. This loss function does not provide a sufficient bias for increasing precision. This also explains why optimizing \( F_1, F_2 \) are more effective than optimizing \( F_{0.5} \) which is precision oriented.

### 8 Conclusion and Future work

We have shown on multiple GED tasks and datasets that optimizing \( F_1 \) and \( F_2 \) outperforms sampling for these evaluation metrics, while maintaining accuracy and precision above what is achieved through sampling methods. Directly optimizing evaluation metrics scores over sampling since: (i) the optimization objective can incorporate precision requirements and needs no empirical determination of hyper-parameters, and (ii) no data loss/corruption due to sampling.

It is beneficial to use \( F_2 \) optimization to learn GED classifiers designed for recall-oriented use-cases like copyediting. The gains are largely due to improvement in recall, and we show the inherent difficulties in optimizing a precision-oriented metric. Future directions of work include improving precision and direct optimization of evaluation metrics for grammatical error correction.

A natural extension is to apply this method to error detection in native speaker text and machine translation output. Our method also has wider applicability to other problems in NLP which encounter the imbalanced dataset problem e.g. sarcasm detection, sentiment thwarting detection, WSD, NER, etc.

### References

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE:
| Method                       | Noun Number | Article | Preposition |
|-----------------------------|-------------|---------|-------------|
|                             | A          | P   | R     | A   | P   | R     | A   | P   | R     |
| Hinge Loss                  | 98.54      | 54.29 | 1.77 | 97.50 | 42.39 | 1.60 | 98.41 | 42.91 | 4.32 |
| Random Undersampling (p=0.3) | 91.79      | 9.20  | 51.88 | 89.44 | 12.33 | 53.06 | 90.92 | 5.65 | 30.54 |
| Cost-Sensitive(p=0.3)       | 95.16      | 9.11  | 25.63 | 94.29 | 16.78 | 32.62 | 98.27 | 24.26 | 4.89 |
| SMOTE (p=1.0)               | 98.16      | 25.00 | 11.07 | 97.34 | 24.15 | 3.32 | 98.38 | 31.46 | 2.88 |
| SVM-Perf F$_1$              | 95.72      | 13.79 | 36.22 | 93.16 | 15.77 | 40.30 | 87.01 | 5.44 | 44.63 |
| SVM-Perf F$_2$              | 94.39      | 11.72 | 43.19 | 93.26 | 15.86 | 39.68 | 86.97 | 5.40 | 44.24 |

Table 6: NUCLE dataset: Comparison of Accuracy (A), Precision (P), Recall (R) (scores in %)

| Method          | Noun Number | Article | Preposition |
|-----------------|-------------|---------|-------------|
|                 |             |         |             |
| SVM-Perf Precision | 12.60  | 39.92 | 15.97 | 39.26 | 7.52  | 0.45  |
| SVM-Perf Recall   | 3.93      | 63.16 | 7.35  | 56.37 | 4.02  | 52.76 |

Table 7: NUCLE dataset: Optimizing Precision and Recall (scores in %)

Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16(1).

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel R Tetreault. 2012. Problems in evaluating grammatical error detection systems. In *COLING*, pages 611–628. Citeseer.

Daniel Dahlmeier and Hwee Tou Ng. 2012. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng. 2012. NUS at the HOO 2012 Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the 8th Workshop on Innovative Use of NLP for Building Educational Applications*.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. 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*.

Pedro Domingos. 1999. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*.

Haibo He and Eduardo A Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9).

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362. Association for Computational Linguistics.

Emi Izumi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2004. The NICT JLE Corpus: Exploiting the Language Learners Speech Database for Research and Education. *International Journal of the Computer, the Internet and Management*.

Thorsten Joachims. Svm-perf: Support vector machine for multivariate performance measures. http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html/.

Thorsten Joachims. Svmlight: Support vector machine. http://svmlight joachims.org/.

Thorsten Joachims. 2005. A Support Vector Method for Multivariate Performance Measures. In *Proceedings of the 22nd International Conference on Machine Learning*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by Data-Intensive and Feature-Rich Statistical Machine Translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Anoop Kunchukuttan, Ritesh Shah, and Pushpak Bhattacharyya. 2013. IITB System for CoNLL 2013 Shared Task: A Hybrid Approach to Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*.

Anoop Kunchukuttan, Sriram Chaudhury, and Pushpak Bhattacharyya. 2014. Tuning a Grammar Correction System for Increased Precision. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.
Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics—Volume 1.

Desmond Darma Putra and Lili Szabo. 2013. UdS at CoNLL 2013 Shared Task. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task.

Alla Rozovskaya and Dan Roth. 2010. Generating confusion sets for context-sensitive error correction. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing.

Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. The UI system in the HOO 2012 shared task on error correction. In Proceedings of the Seventh Workshop on Building Educational Applications Using NLP.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. In Journal of Machine Learning Research.

Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In Proceedings of the 24th International Conference on Machine learning.

Junwen Xing, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Xiaodong Zeng. 2013. UM-Checker: A Hybrid System for English Grammatical Error Correction. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task.

Bianca Zadrozny, John Langford, and Naoki Abe. 2003. Cost-sensitive learning by cost-proportionate example weighting. In Third IEEE International Conference on Data Mining.