Boosting Text Augmentation via Hybrid Instance Filtering Framework

Heng Yang, Ke Li*
Department of Computer Science, University of Exeter, EX4 4QF, Exeter, UK
{hy345, k.li}@exeter.ac.uk

Abstract

Text augmentation is an effective technique for addressing the problem of insufficient data in natural language processing. However, existing text augmentation methods tend to focus on few-shot scenarios and usually perform poorly on large public datasets. Our research indicates that existing augmentation methods often generate instances with shifted feature spaces, which leads to a drop in performance on the augmented data (for example, EDA generally loses $\approx 2\%$ in aspect-based sentiment classification). To address this problem, we propose a hybrid instance-filtering framework (BoostAug) based on pre-trained language models that can maintain a similar feature space with natural datasets. BoostAug is transferable to existing text augmentation methods (such as synonym substitution and back translation) and significantly improves the augmentation performance by $\approx 2-3\%$ in classification accuracy. Our experimental results on three classification tasks and nine public datasets show that BoostAug addresses the performance drop problem and outperforms state-of-the-art text augmentation methods. Additionally, we release the code to help improve existing augmentation methods on large datasets.

1 Introduction

Recent pre-trained language models (PLMs) (Devlin et al., 2019; Brown et al., 2020; He et al., 2021; Yoo et al., 2021) have been able to learn from large amounts of text data. However, this also leads to a critical problem of data insufficiency in many low-resource fine-tuning scenarios (Chen et al., 2020; Zhou et al., 2022a; Miao et al., 2021; Kim et al., 2022; Wang et al., 2022b; Yang et al., 2022). Despite this, existing augmentation studies still encounter failures on large public datasets. While some studies(Ng et al., 2020; Body et al., 2021; Chang et al., 2021; Luo et al., 2021) have attempted to leverage the language modeling capabilities of PLMs in text augmentation, these methods still suffer from performance drops on large datasets.

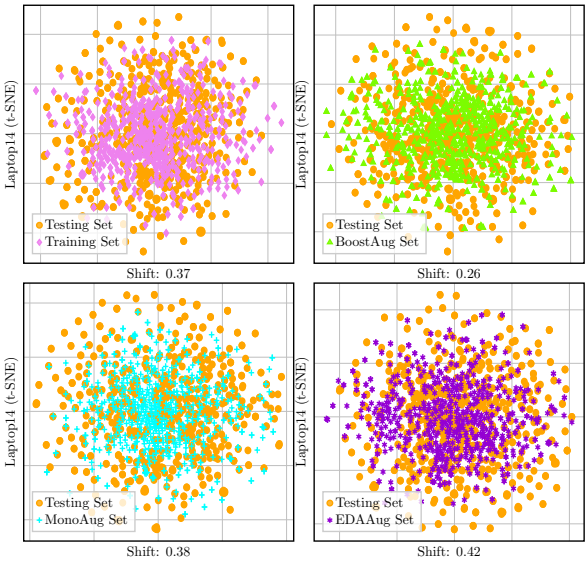

Figure 1: The visualization of feature space shift of the Laptop14 dataset based on t-SNE. We calculate the shift metric $\mathcal{S}$ of feature space between augmented and natural instances. The augmentation methods are BoostAug, MonoAug, and EDA augmentation, respectively. Our BoostAug has the least feature space shift.

To explore the root cause of this failure mode, we conducted experiments to explain the difference between "good" and "bad" augmentation instances. Our study found that existing augmentation methods (Wei and Zou, 2019; Coulombe, 2018; Li et al., 2019; Kumar et al., 2019; Ng et al., 2020) usually fail to maintain the feature space in augmentation instances, which leads to bad instances. This shift in feature space occurs in both edit-based and PLM-based augmentation methods. For example, edit-based methods can introduce breaking changes that corrupt the meaning of the text, while PLM-based methods can introduce out-of-vocabulary words. In particular, for the edit-

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2023, pages 1652–1669
July 9-14, 2023

based methods, the shifted feature space mainly comes from breaking text transformations, such as changing important words (e.g., 'but ') in sentiment analysis. As for PLM-based methods, they usually introduce out-of-vocabulary words due to word substitution and insertion, which leads to an adverse meaning in sentiment analysis tasks.

To address the performance drop in existing augmentation methods caused by shifted feature space, we propose a hybrid instance-filtering framework (BOOSTAUG) based on PLMs to guide augmentation instance generation. Unlike other existing methods (Kumar et al., 2020), we use PLMs as a powerful instance filter to maintain the feature space, rather than as an augmentor. This is based on our finding that PLMs fine-tuned on natural datasets are familiar with the identical feature space distribution. The proposed framework consists of four instance filtering strategies: perplexity filtering, confidence ranking, predicted label constraint, and a cross-boosting strategy. These strategies are discussed in more detail in section Section 2.3. Compared to prominent studies, BOOSTAUG is a pure instance-filtering framework that can improve the performance of existing text augmentation methods by maintaining the feature space.

With the mitigation of feature space shift, BOOSTAUG can generate more valid augmentation instances and improve existing augmentation methods' performance, which more augmentation instances generally trigger performance sacrifice in other studies (Coulombe, 2018; Wei and Zou, 2019; Li et al., 2019; Kumar et al., 2020)). According to our experimental results on three fine-grained and coarse-grained text classification tasks, BOOSTAUG[1] significantly alleviates feature space shifts for existing augmentation methods.

Our main contributions are:

- We propose the feature space shift to explain the performance drop in existing text augmentation methods, which is ubiquitous in full dataset augmentation scenarios.
- We propose a universal augmentation instance filter framework to mitigate feature space shift and significantly improve the performance on the ABSC and TC tasks.
- Our experiments show that the existing text augmentation methods can be easily improved by employing BOOSTAUG.

[1]We release the source code and experiment scripts of BOOSTAUG at: https://github.com/yangheng95/BoostTextAugmentation.

Algorithm 1: The pseudo code of BOOSTAUG

1 Split $\mathcal{D}$ into k folds, $\mathcal{D} := \{\mathcal{F}^i\}_{i=1}^{k}$;
2 $\mathcal{D}_{\text{aug}} := \emptyset$;
3 **for** $i \leftarrow 1$ **to** k **do**
4 $\mathcal{D}^i_{\text{aug}} := \emptyset$, $\mathcal{D}^i_{\text{boost}} := \mathcal{F}^i$;
5 Randomly pick up $k-2$ folds except $\mathcal{F}^i$ to constitute $\mathcal{D}^i_{\text{train}}$;
6 $\mathcal{D}^i_{\text{valid}} := \mathcal{F} \setminus (\mathcal{F}^i \bigcup \mathcal{D}^i_{\text{train}})$;
7 Use the DeBERTa on $\mathcal{D}^i_{\text{train}}$ and $\mathcal{D}^i_{\text{valid}}$ to build the surrogate language model;
8 **forall** $d_{\text{org}} \in \mathcal{D}^i_{\text{boost}}$ **do**
9 $\mathcal{D}^i_{\text{aug}} := F(d^i_{\text{org}}, \tilde{N}, \Theta)$;
10 **forall** $d_{\text{aug}} \in \mathcal{D}^i_{\text{aug}}$ **do**
11 Use the surrogate language model to predict $\mathbb{P}(d_{\text{aug}})$, $\mathbb{C}(d_{\text{aug}})$, and the $\tilde{\ell}_{\text{aug}}$ of d_{aug};
12 **if** $\mathbb{P}(d_{\text{aug}}) \geq \alpha \parallel \mathbb{C}(d_{\text{aug}}) \leq \beta \parallel \tilde{\ell}_{d_{\text{aug}}} \neq \tilde{\ell}_{d_{\text{org}}}$ **then**
13 $\mathcal{D}^i_{\text{aug}} := \mathcal{D}^i_{\text{aug}} \setminus \{d_{\text{aug}}\}$;
14 $\mathcal{D}_{\text{aug}} := \mathcal{D}_{\text{aug}} \bigcup \mathcal{D}^i_{\text{aug}}$;
15 $\mathcal{D}_{\text{aug}} := \mathcal{D}_{\text{aug}} \bigcup \mathcal{D}^i_{\text{boost}}$;
16 **return** $\mathcal{D}_{\text{aug}}$

2 Proposed Method

The workflow of BOOSTAUG is shown in Figure 2 and the pseudo code is given in Algorithm 1. Different from most existing studies, which focus on unsupervised instance generation, BOOSTAUG serves as an instance filter to improve existing augmentation methods. The framework consists of two main phases: 1) Phase #1: the training of surrogate language models; 2) Phase #2: surrogate language models guided augmentation instance filtering. The following paragraphs will provide a detailed explanation of each step of the implementation.

2.1 Surrogate Language Model Training

At the beginning of Phase #1, the original training dataset is divided into $k \geq 3$ folds where the $k-2$ ones are used for training (denoted as the training fold) while the other two are used for the validation and augmentation purposes, denoted as the validation and boosting fold, respectively[2] (lines 4-6). Note that the generated augmentation instances, which will be introduced in Section 2.2, can be

[2]We iteratively select the i-th fold, $i \in 1, \cdots, k$, as the boosting fold (line 3 in Algorithm 1). The validation fold is used to select the best checkpoint of the surrogate language model to filter the augmented instances. This process is repeated k times to ensure that all the folds have been used for validation and boosting at least once, thus avoiding data overlapping between the training and validation folds.

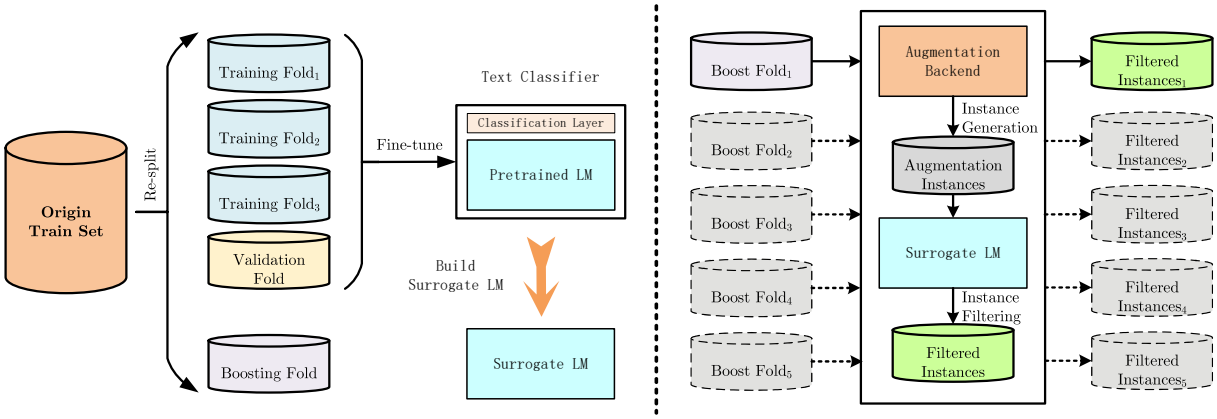

Figure 2: The workflow of BOOSTAUG can be divided into two phases: Phase #1 and Phase #2. In Phase #1, we fine-tune a DeBERTa-based classification model using re-split training and validation sets and extract the fine-tuned DeBERTa to build a surrogate language model. In Phase #2, BOOSTAUG employs a text augmentation backend to generate raw augmentations and filters out low-quality instances identified by the surrogate language model. To avoid data overlapping between the training folds and validation fold, BOOSTAUG performs k-fold cross-boosting, meaning that Phase #1 and #2 are repeated k times.

identical to the instances in training folds the surrogate language model. This data overlapping problem will lead to a shifted feature space. We argue that the proposed k-fold augmentation approach, a.k.a. "cross-boosting", can alleviate the feature space shift of the augmentation instances, which will be validated and discussed in detail in Section 4.3. The main crux of Phase #1 is to build a surrogate language model as a filter to guide the elimination of harmful and poor augmentation instances.

We construct a temporary classification model using the DeBERTa (He et al., 2021) architecture. This model is then fine-tuned using the data in the $k-2$ training folds and the validation fold to capture the semantic features present in the data (line 7). It is important to note that we do not use the original training dataset for this fine-tuning process. Once the fine-tuning is complete, the language model constructed from the DeBERTa classification model is then utilized as the surrogate language model in the instance filtering step in Phase #2 of BOOSTAUG.

This is different from existing works that use a pre-trained language model to directly generate augmentation instances. We clarify our motivation for this from the following two aspects.

- In addition to modeling the semantic feature, the surrogate language model can provide more information that can be useful for the quality control of the augmentation instances, such as text perplexity, classification confidence, and predicted label.
- Compared to the instance generation, we argue that the instance filtering approach can be readily integrated with any existing text augmentation approach.

2.2 Augmentation Instance Generation

As a building block of Phase #2, we apply some prevalent data augmentation approaches as the back end to generate the augmentation instances in BOOSTAUG (line 9). More specifically, let $\mathcal{D}_{\text{org}} := \{d^i_{\text{org}}\}_{i=1}^N$ be the original training dataset. $d^i_{\text{org}} := \langle s^i, \ell^i \rangle$ is a data instance where s^i indicates a sentence and ℓ^i is the corresponding label, $i \in 1, \cdots, N$. By applying the transformation function $F(\cdot,\cdot,\cdot)$ upon d^i_{org} as follows, we expect to obtain a set of augmentation instances $\mathcal{D}^i_{\text{aug}}$ for d^i_{org}:

$$\mathcal{D}^i_{\text{aug}} := F(d^i_{\text{org}}, \tilde{N}, \Theta), \quad (1)$$

where $\tilde{N} \geq 1$ is used to control the maximum number of generated augmentation instances. In the end, the final augmentation set is constituted as $\mathcal{D}_{\text{aug}} := \bigcup_{i=1}^N \mathcal{D}^i_{\text{aug}}$ (line 14). Note that depending on the specific augmentation back end, there can be more than one strategy to constitute the transformation function. For example, EDA (Wei and Zou, 2019) has four transformation strategies, including synonym replacement, random insertion, random swap, and random deletion. Θ consists

of the parameters associated with the transformation strategies of the augmentation back end, e.g., the percentage of words to be modified and the mutation probability of a word.

2.3 Instance Filtering

Our preliminary experiments have shown that merely using data augmentation can be detrimental to the modeling performance, no matter how many augmentation instances are applied in the training process. In addition, our experiments in Section 4.3 have shown a surprising feature space shift between the original data and the augmented instances in the feature space. To mitigate this issue, BOOSTAUG proposes an instance filtering approach to control the quality of the augmentation instances. It consists of three filtering strategies, including perplexity filtering, confidence ranking, and predicted label constraint, which will be delineated in the following paragraphs, respectively. Note that all these filtering strategies are built on the surrogate language model developed in `Phase` #1 of BOOSTAUG (lines 12 and 13).

2.3.1 Perplexity Filtering

Text perplexity is a widely used metric to evaluate the modeling capability of a language model (Chen and Goodman, 1999; Sennrich, 2012). Our preliminary experiments have shown that low-quality instances have a relatively high perplexity. This indicates that perplexity information can be used to evaluate the quality of an augmentation instance. Since the surrogate language model built in `Phase` #1 is bidirectional, the text perplexity of an augmentation instance d_{aug} is calculated as:

$$\mathbb{P}(d_{\text{aug}}) = \prod_{i=1}^{s} p\left(w_i \mid w_1, \cdots, w_{i-1}, w_{i+1}, \cdots, w_s\right), \tag{2}$$

where w_i represents the token in the context. s is the number of tokens in d_{aug} and $p\left(w_i \mid w_1, \cdots, w_{i-1}, w_{i+1}, \cdots, w_s\right)$ is the probability of w_i conditioned on the preceding tokens, according to the surrogate language model, $i \in 1, \cdots, s$. Note that d_{aug} is treated as a low-quality instance and is discarded if $\mathbb{P}(d_{\text{aug}}) \geq \alpha$ while $\alpha \geq 0$ is a predefined threshold.

2.3.2 Confidence Ranking

We observe a significant feature space shift in the augmentation instances. These instances will be allocated with low confidences by the surrogate language model. In this case, we can leverage the classification confidence as a driver to control the quality of the augmentation instances. However, it is natural that long texts can have way more augmentation instances than short texts, thus leading to the so-called unbalanced distribution. Besides, the confidence of most augmentation instances is $\geq 95\%$, which is not selective as the criterion for instance filtering. To mitigate the unbalanced distribution in augmentation instances and make use of confidence, we develop a confidence ranking strategy to eliminate the redundant augmentation instances generated from long texts while retaining the rare instances having a relatively low confidence. More specifically, we apply a softmax operation on the output hidden state learned by the surrogate language model, denoted as $\mathbb{H}(d_{\text{aug}})$, to evaluate the confidence of d_{aug} as:

$$\mathbb{C}(d_{\text{aug}}) = \text{argmax}\left(\frac{\exp(\mathbb{H}_{d\text{aug}})}{\sum_1^c \exp(\mathbb{H}_{d\text{aug}})}\right), \tag{3}$$

where c is the number of classes in the original training dataset. To conduct the confidence ranking, $2 \times \tilde{N}$ instances are generated at first, while only the top $\tilde{N}$ instances are selected to carry out the confidence ranking. By doing so, we expect to obtain a balanced augmentation dataset even when there is a large variance in the confidence predicted by the surrogate language model. After the confidence ranking, the augmentation instances with $\mathcal{C}_{d_{\text{aug}}} \leq \beta$ are discarded while $\beta \geq 0$ is a fixed threshold.

2.3.3 Predicted Label Constraint

Due to some breaking text transformation, text augmentation can lead to noisy data, e.g., changing a word "greatest" to "worst" in a sentence leads to an adverse label in a sentiment analysis task. Since the surrogate language model can predict the label of an augmentation instance based on its confidence distribution, we develop another filtering strategy that eliminates the augmentation instances whose predicted label $\tilde{\ell}_{d_{\text{aug}}}$ is different from the ground truth. By doing so, we expect to mitigate the feature space bias.

2.4 Feature Space Shift Metric

To quantify the shift of the feature space, we propose an ensemble metric based on the overlapping ratio and distribution skewness of the t-SNE-based augmented instances' feature space.

The feature space overlapping ratio measures the diversity of the augmented instances. A larger overlapping ratio indicates that more natural instances have corresponding augmented instances. On the other hand, the distribution skewness measure describes the uniformity of the distribution of the augmented instances. A smaller distribution skewness indicates that the natural instances have approximately equal numbers of corresponding augmented instances. To calculate the feature space shift, we first calculate the overlapping ratio and distribution skewness of the natural instances and their corresponding augmented instances. The feature space shift is calculated as follows:

$$\mathcal{S} = 1 - \mathcal{O} + sk, \tag{4}$$

where $\mathcal{O}$ and sk are the feature space convex hull overlapping ratio and feature space distribution skewness, which will be introduced in the following subsections.

2.4.1 Convex hull overlapping calculation

To calculate the convex hull overlapping rate, we use the Graham Scan algorithm[3] (Graham, 1972) to find the convex hulls for the test set and target dataset in the t-SNE visualization, respectively.

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ represent the convex hulls of two datasets in the t-SNE visualization; we calculate the overlapping rate as follows:

$$\mathcal{O} = \frac{\mathcal{P}_1 \cap \mathcal{P}_2}{\mathcal{P}_1 \cup \mathcal{P}_2}, \tag{5}$$

where $\cap$ and $\cup$ denote convex hull intersection and union operations, respectively. $\mathcal{O}$ is the overlapping rate between $\mathcal{P}_1$ and $\mathcal{P}_2$.

2.4.2 Distribution skewness calculation

The skewness of an example distribution is computed as follows:

$$sk = \frac{m_3}{m_2^{3/2}}, \tag{6}$$

$$m_i = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^i, \tag{7}$$

where N is the number of instances in the distribution; sk is the skewness of an example distribution. m_i and $\bar{x}$ are the i-th central moment and mean of the example distribution, respectively. Because the t-SNE has two dimensions (namely x and y axes), we measure the global skewness of the target dataset (e.g., training set, augmentation set) by summarizing the absolute value of skewness on the x and y axes in t-SNE:

$$sk^g = |sk^x| + |sk^y|, \tag{8}$$

where sk^g is the global skewness of the target dataset; sk^x and sk^y are the skewness on the x and y axes, respectively.

By combining the convex hull overlapping ratio and distribution skewness, the proposed feature space shift metric offers a comprehensive view of how well the augmented instances align with the original data distribution. This metric can be used to evaluate the effectiveness of different data augmentation approaches, as well as to inform the fine-tuning process for better model performance.

3 Experimental Setup

3.1 Datasets

Our experiments are conducted on three classification tasks: the sentence-level text classification (TC), the aspect-based sentiment classification (ABSC), and natural language inference (NLI). The datasets used for the TC task include SST2, SST5 (Socher et al., 2013) from the Stanford Sentiment Treebank, and AGNews10K[4](Zhang et al., 2015). Meanwhile, the datasets used for the ABSC task are Laptop14, Restaurant14(Pontiki et al., 2014), Restaurant15 (Pontiki et al., 2015), Restaurant16 (Pontiki et al., 2016), and MAMS (Jiang et al., 2019). The datasets[5] used for the NLI task are the SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) datasets, respectively. The split of these datasets is summarized in Table 1. The commonly used `Accuracy` (i.e., `Acc`) and macro `F1` are used as the metrics for evaluating the performance of different algorithms following existing research (Wang et al., 2016; Zhou et al., 2022a). Additionally, all experiments are repeated five times with different random seeds. Detailed information on the hyper-parameter settings and sensitivity tests of α and β can be found in Appendix A.

[3] https://github.com/shapely/shapely.

[4] We use the first $10,000$ examples to build the AGNews10K dataset ($7,000$ for training, $1,000$ for validation, and $2,000$ for testing), which is large enough compared to other datasets.

[5] We select the first 1000 training examples as the training set and keep the original validation/testing sets for experimental efficiency.

Table 1: The summary of experimental datasets for the text classification, aspect-based sentiment analysis and natural language inference tasks.

Dataset	Training Set	Validation Set	Testing Set
Laptop14	2328	0	638
Restaurant14	3608	0	1120
Restaurant15	1120	0	540
Restaurant16	1746	0	615
MAMS	11186	1332	1336
SST2	6920	872	1821
SST5	8544	1101	2210
AGNews10K	7000	1000	2000
SNLI	1000	10000	10000
MNLI	1000	20000	0

3.2 Augment Backends

We use BoostAug to improve five state-of-the-art baseline text augmentation methods, all of which are used as the text augmentation backend of BoostAug. Please find the introductions of these baselines in Appendix B and refer to Table 6 for detailed performance of BoostAug based on different backends.

We also compare BoostAug enhanced EDA with the following text augmentation methods:

- EDA (TextAttack[6]) (Wei and Zou, 2019) performs text augmentation via random word insertions, substitutions, and deletions.
- SynonymAug (NLPAug[7]) (Niu and Bansal, 2018) replaces words in the original text with their synonyms. This method has been shown to be effective in improving the robustness of models on certain tasks.
- TAA (Ren et al., 2021) is a Bayesian optimization-based text augmentation method. It searches augmentation policies and automatically finds the best augmentation instances.
- AEDA (Karimi et al., 2021) is based on the EDA, which attempts to maintain the order of the words while changing their positions in the context. Besides, it alleviates breaking changes such as critical deletions and improves the robustness.
- AMDA (Si et al., 2021) linearly interpolates the representations of pairs of training instances, which has a diversified augmentation set compared to discrete text adversarial augmentation.

In our experiments, LSTM, BERT-BASE(Devlin et al., 2019), and DeBERTa-BASE(He et al., 2021) are used as the objective models for the TC task. FastLCF is an objective model available for the ABSC task.

[6] https://github.com/QData/TextAttack
[7] https://github.com/makcedward/nlpaug

4 Experimental Results

4.1 Main Results

From the results shown in Table 2, it is clear that BoostAug consistently improves the performance of the text augmentation method EDA across all datasets and models. It is also worth noting that some traditional text augmentation methods can actually harm the performance of the classification models. Additionally, the performance improvement is relatively small for larger datasets like SST-2, SST-5, and MAMS. Furthermore, the performance of LSTM is more affected by text augmentation, as it lacks the knowledge gained from the large-scale corpus that is available in PLMs.

When comparing the different text augmentation methods, it is apparent that EDA performs the best, despite being the simplest method. On the other hand, SplitAug performs the worst for LSTM because its augmentation instances are heavily biased in the feature space due to the word splitting transformation. The performance of SpellingAug is similar to EDA. This can be attributed to the fact that PLMs have already captured some common misspellings during pre-training. Additionally, PLM-based augmentation methods like WordsEmbsAug tend to generate instances with unknown words, further exacerbating the feature space shift of the augmented texts.

We also compare the performance of BoostAug with several state-of-the-art text augmentation methods. The results of these comparisons can be found in Table 3. From the results, it can be seen that even when using EDA (Wei and Zou, 2019) as the backend, BoostAug outperforms other state-of-the-art methods such as AEDA (Karimi et al., 2021), AMDA (Si et al., 2021), and Bayesian optimization-based TAA (Ren et al., 2021) on the full SST2 dataset.

4.2 Ablation Study

To gain a deeper understanding of the working mechanism of BoostAug, we conduct experiments to evaluate the effectiveness of cross-boosting, predicted label constraint, confidence ranking, and perplexity filtering. The results, which can be found in Table 4, show that the performance of the variant MonoAug is significantly lower than that of BoostAug. This is because MonoAug trains the surrogate language model using the entire

Table 2: The performance comparison between BOOSTAUG-enhanced EDA and baseline augmentation methods. The best and second-best metric values are highlighted in **bold** and underlined faces, respectively. None indicates the vanilla version without using text augmentation. † indicates that BOOSTAUG is significantly better than the backend according to the Wilcoxon's rank sum test at a 0.05 significance level. "-" indicates that FastLCF is not available for text classification or the results are not considered due to resource limitation.

Augmentation	Model	Laptop14		Restaurant14		Restaurant15		Restaurant16		MAMS		SST2		SST5		AGNews10K	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
None	LSTM	71.32	65.45	77.54	66.89	78.61	58.54	87.40	64.41	56.96	56.18	84.36	84.36	45.29	44.61	87.60	87.36
	BERT	79.47	75.70	85.18	78.31	83.61	69.73	91.3	77.16	82.78	82.04	90.88	90.88	53.53	52.06	92.47	92.26
	DeBERTa	83.31	80.02	87.72	81.73	86.58	74.22	93.01	81.93	83.31	82.87	95.07	95.07	56.47	55.58	92.30	92.13
	FastLCF	83.23	79.68	88.5	82.7	87.74	73.69	93.69	81.66	83.46	82.88	-	-	-	-	-	-
EDA	LSTM	68.65	62.09	76.18	62.41	76.30	56.88	85.59	61.78	56.59	55.33	84.79	84.79	43.85	43.85	87.72	87.46
	BERT	78.37	74.23	83.75	75.38	81.85	65.63	91.38	77.27	81.81	81.10	91.16	91.16	51.58	50.49	92.50	92.28
	DeBERTa	80.96	78.65	86.79	79.82	84.44	70.40	93.01	77.59	81.96	81.96	94.07	94.07	56.43	53.88	92.55	92.33
	FastLCF	81.97	79.57	87.68	81.52	86.39	72.51	93.17	78.96	82.19	81.63	-	-	-	-	-	-
SpellingAug	LSTM	67.24	60.30	75.36	63.01	73.52	49.04	84.72	53.92	55.99	55.16	83.14	83.14	41.45	40.40	87.25	86.96
	BERT	73.59	69.11	82.54	73.18	79.63	62.32	89.76	74.74	81.89	81.42	91.00	91.00	52.26	50.90	92.42	92.22
	DeBERTa	80.17	76.01	85.13	76.67	85.83	71.54	92.76	78.33	81.89	81.24	93.68	93.68	55.95	53.78	92.68	92.50
	FastLCF	79.62	74.81	86.03	78.73	87.41	75.14	92.60	75.27	82.19	81.66	-	-	-	-	-	-
SplitAug	LSTM	62.98	56.53	73.43	58.57	70.19	45.71	83.93	54.41	56.74	55.34	84.29	84.29	44.00	42.10	87.23	87.01
	BERT	75.47	70.56	82.86	74.48	82.87	65.19	90.98	77.51	81.74	81.35	90.88	90.88	51.99	50.95	92.45	92.16
	DeBERTa	79.15	75.72	86.03	79.28	85.46	70.43	92.76	79.79	81.59	81.09	94.29	94.29	55.51	49.77	92.52	92.29
	FastLCF	81.82	78.46	86.34	78.36	86.67	70.87	93.09	76.50	82.07	81.53	-	-	-	-	-	-
ContextualWordEmbsAug	LSTM	67.40	61.57	75.62	62.13	74.44	51.67	84.98	58.67	56.06	55.10	83.14	83.14	44.07	42.03	87.53	87.24
	BERT	75.63	70.79	83.26	75.11	78.61	61.48	90.24	72.37	81.29	80.50	91.02	91.02	51.27	50.27	92.10	91.86
	DeBERTa	76.88	71.98	85.49	77.22	84.63	70.50	92.28	77.42	81.66	81.32	94.12	94.12	55.48	53.60	92.80	92.62
	FastLCF	79.08	74.61	85.62	76.88	84.91	70.06	91.38	76.27	81.89	81.09	-	-	-	-	-	-
BackTranslationAug	LSTM	68.50	62.22	78.12	66.70	78.85	59.08	86.97	63.47	-	-	-	-	-	-	-	-
	BERT	79.94	76.19	85.54	78.51	84.42	72.05	92.02	85.78	-	-	-	-	-	-	-	-
	DeBERTa	84.17	81.15	88.93	83.54	89.42	78.67	93.97	80.52	-	-	-	-	-	-	-	-
	FastLCF	82.76	79.82	89.46	84.94	88.13	75.70	94.14	81.82	-	-	-	-	-	-	-	-
BOOSTAUG (EDA)	LSTM	73.20†	67.46†	79.12†	68.07†	80.06†	59.61†	87.80†	65.33†	59.21†	59.58†	85.83†	85.83†	45.93†	43.59†	88.45	88.16
	BERT	80.10†	76.48†	86.34†	79.99†	86.12†	73.79†	91.95†	79.12†	84.01†	83.44†	92.33†	92.33†	53.94†	52.80†	92.48	92.25
	DeBERTa	84.56†	81.77†	89.02†	83.35†	88.33†	76.77†	93.58†	81.93†	**84.51**†	**83.97**†	**96.09**†	**96.09**†	**57.78**†	**56.15**†	**92.95**	**92.76**
	FastLCF	**85.11**†	**82.18**†	**90.38**†	**85.04**†	**89.81**†	**77.92**†	**94.37**†	**82.67**†	84.13†	82.97†	-	-	-	-	-	-

Table 3: The performance comparison on augmented **SST2** dataset between different augmentation methods. We list the standard deviations for each method, while "-" indicates the standard deviation is not available. * is derived from our experiments.

Augmentation	Model	Acc	F1
None*	BERT	90.88 (0.31)	90.87 (0.31)
EDA*	BERT	90.99 (0.46)	90.99 (0.46)
SynonymAug*	BERT	91.32 (0.55)	91.31 (0.55)
TAA*	BERT	90.94 (0.31)	90.94 (0.31)
AEDA	BERT	91.76 (-)	—
AMDA	BERT	91.54 (-)	—
BOOSTAUG(EDA)	BERT	**92.33** (0.29)	**92.33** (0.29)

training set, leading to a high degree of similarity between the original and augmentation instances. This data overlapping problem, as discussed in Section 2.1, results in biased instance filtering and overfitting of the instances to the training fold data distribution. Additionally, the variant without the perplexity filtering strategy performs the worst, indicating that the perplexity filtering strategy is crucial in removing instances with syntactical and grammatical errors. The performance of the variants without the predicted label constraint and confidence ranking is similar, with the label constraint helping to prevent the mutation of features into an adverse meaning and the confidence ranking helping to eliminate out-of-domain words and reduce feature space shift.

4.3 Feature Space Shift Investigation

In this subsection, we explore the feature space shift problem in more detail by using visualizations and the feature space shift metric. We use t-SNE to visualize the distribution of the features of the testing set and compare it to different augmented variants. The full results of feature space shift metrics are available in Figure 6. The results of feature space shift metrics in our experiment show that the augmentation instances generated by BOOSTAUG have the least shift of feature space. Specifically, the overlapping ratio and skewness in relation to the testing set are consistently better than those of the training set. This explains the performance improvement seen when using BOOSTAUG in previous experiments. In contrast, the augmentation instances generated by EDA, which was the best peer text augmentation method, have a worse overlapping rate compared to even the training set. This explains the performance degradation when using EDA on the baseline classification models. It is also noteworthy that the quality of the augmentation instances generated by MonoAug is better than EDA.

4.4 Effect of Augmentation Instances Number

To further understand the effectiveness of BOOSTAUG, we conduct an experiment to analyze the relationship between the number of augmentation instances generated and the performance of

Table 4: The performance comparison between different ablated variants of BoostAug.

Ablation	Model	MAMS		SST2		SST5		AGNews10K	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
BoostAug(EDA)	LSTM	59.21	59.58	85.77	85.77	45.79	43.84	88.45	88.16
	BERT	84.01	83.44	92.33	92.33	52.38	51.70	92.48	92.25
	DeBERTa	**84.51**	**83.97**	**96.09**	**96.09**	57.78	56.15	**92.95**	**92.76**
MonoAug	LSTM	57.26	55.70	84.62	84.62	45.25	42.91	87.55	87.32
	BERT	83.68	83.04	91.16	91.16	52.90	52.12	92.40	92.19
	DeBERTa	83.38	82.87	94.18	94.18	56.92	55.81	92.32	92.09
w/o Confidence	LSTM	56.19	55.94	85.08	85.08	45.48	44.89	86.98	86.61
	BERT	83.26	82.62	91.16	91.16	53.21	52.27	92.20	92.00
	DeBERTa	83.47	82.08	95.22	95.22	57.10	55.97	92.93	92.75
w/o Perplexity	LSTM	55.54	55.46	85.67	85.67	46.47	43.25	87.40	86.99
	BERT	83.16	82.58	92.04	92.04	52.67	51.02	92.50	92.30
	DeBERTa	83.53	83.04	95.39	95.39	**58.10**	**56.78**	92.60	92.36
w/o Label Constraint	LSTM	56.06	55.00	84.90	84.88	44.75	43.44	86.55	86.23
	BERT	83.01	82.57	92.04	92.03	52.58	51.33	91.80	91.60
	DeBERTa	82.41	82.01	95.06	95.06	56.70	54.91	92.85	92.60

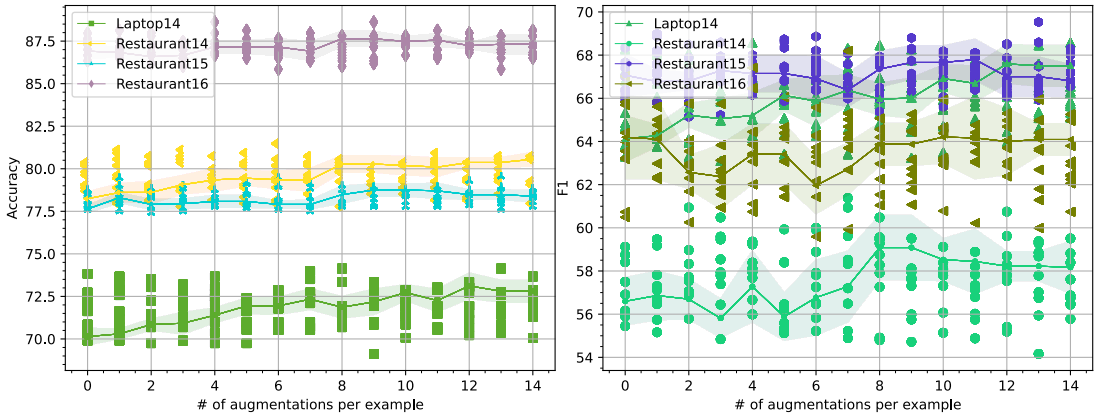

Figure 3: Trajectories of the Acc and the F1 values with error bars versus the number of augmentation instances generated for an example by using BoostAug(EDA). The trajectory visualization plot of MonoAug and EDA can be found in Figure 7

the classification models. We use Acc and F1 as the evaluation metrics and plot the trajectories of these metrics with error bars against the number of augmentation instances generated for an example by using BoostAug. The results are shown in Figure 3. For comparison, the trajectory visualization plots of MonoAug and EDA can also be found in Figure 7. From the results, it is clear to see that the performance of the classification models improves as the number of augmentation instances increases, but eventually reaches a saturation point. Furthermore, it is observed that the performance improvement achieved by BoostAug is consistently better than that of MonoAug and EDA. This further confirms the effectiveness of BoostAug in mitigating the feature space shift problem and improving the performance of the classification models.

However, it is also important to consider the computational budgets required to generate a large number of augmentation instances, as this can impact the overall efficiency of the text augmentation method being used.

4.5 Hyper-parameter Sensitivity Analysis

We find that there is no single best setting for the two hyper-parameters, α and β, in different situations such as different datasets and backend augmentation methods. To explore the sensitivity of these hyper-parameters, we conducted experiments on the Laptop14 and Restaurant14 datasets and show the Scott-Knott rank test (Mittas and Angelis, 2013) plots and performance box plots in Figure 4 and Figure 5, respectively. We found that the best value of α highly depends on the dataset. For the Laptop14 and Restaurant14 datasets, a value of $\alpha = 0.5$ was found to be the best choice according to Figure 4. However, it's worth noting that the smaller the value of α, the higher the computation complexity due to the need for more

augmentation instances. To balance efficiency and performance, we recommend a value of $\alpha = 0.99$ ($\alpha = 1$ means no augmentation instances survive) in BoostAug, which reduces computation complexity. Additionally, we found that β is relatively easy to determine, with a value of $\beta = 4$ being commonly used.

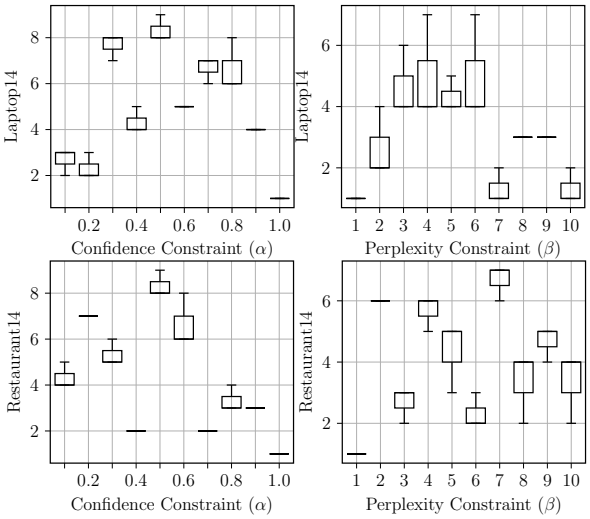

Figure 4: The Scott-knott rank test plots under different α and β in BoostAug(EDA). The bigger rank means better performance.

5 Related Works

As pretraining has advanced, text augmentation techniques have become an increasingly popular area of research (Sennrich et al., 2016; Coulombe, 2018; Li et al., 2019; Wei and Zou, 2019; Kumar et al., 2020; Lewis et al., 2020; Xie et al., 2020; Bi et al., 2021; Ren et al., 2021; Haralabopoulos et al., 2021; Wang et al., 2022c; Yue et al., 2022; Zhou et al., 2022a; Kamalloo et al., 2022; Wang et al., 2022a). Many of these techniques focus on low-resource scenarios (Chen et al., 2020; Zhou et al., 2022a; Kim et al., 2022; Zhou et al., 2022b; Wu et al., 2022; Yang et al., 2022; Wang et al., 2022b; Yang et al., 2022). However, they tend to fail when applied to large public datasets (Zhou et al., 2022a). Recent prominent works (Sennrich et al., 2016; Kumar et al., 2020; Lewis et al., 2020; Ng et al., 2020; Body et al., 2021; Chang et al., 2021; Luo et al., 2021; Wang et al., 2022b) recognize the significance of pre-trained language models (PLMs) for text augmentation and propose PLM-based methods to improve text augmentation. However, the quality of augmentation instances generated by unsupervised PLMs cannot be guaranteed. Some research (Dong et al., 2021) has attempted to use adversarial training in text augmentation, which can improve robustness, but these methods are more suitable for low-sample augmentation scenarios and cause shifted feature spaces in large datasets.

While recent studies have emphasized the importance of quality control for augmentation instances (Lewis et al., 2021; Kamalloo et al., 2022; Wang et al., 2022b), there remains a need for a transferable augmentation instance-filtering framework that can serve as an external quality controller to improve existing text augmentation methods.

Our work aims to address the failure mode of large dataset augmentation and improve existing augmentation methods more widely. Specifically, BoostAug is a simple but effective framework that can work with a variety of existing augmentation backends, including EDA (Wei and Zou, 2019) and PLM-based augmentation (Kumar et al., 2020).

6 Conclusion

Existing text augmentation methods usually lead to performance degeneration in large datasets due to numerous low-quality augmentation instances, while the reason for performance degeneration has not been well explained. We find low-quality augmentation instances usually have shifted feature space compare to natural instances. Therefore, we propose a universal augmentation instance filter framework BoostAug to widely enhance existing text augmentation methods. BoostAug is an external and flexible framework, all the existing text augmentation methods can be seamless improved. Experimental results on three TC datasets and five ABSC datasets show that BoostAug is able to alleviate feature space shift in augmentation instances and significantly improve existing augmentation methods.

Acknowledgements

This work was supported by UKRI Future Leaders Fellowship (MR/X011135/1, MR/S017062/1), NSFC (62076056), Alan Turing Fellowship, EPSRC (2404317), Royal Society (IES/R2/212077) and Amazon Research Award.

7 Limitations

We propose and solve the feature space shift problem in text augmentation. However, there is a limitation that remains. BoostAug cannot preserve

the grammar and syntax to a certain extent. We apply the perplexity filtering strategy, but it is an implicit constraint and cannot ensure the syntax quality of the augmentation instances due to some breaking transformations, such as keyword deletions and modifications. However, we do not need precise grammar and syntax information in most classification tasks, especially in PLM-based classification. For some syntax-sensitive tasks, e.g., syntax parsing and the syntax-based ABSC (Zhang et al., 2019; Phan and Ogunbona, 2020; Dai et al., 2021), ensuring the syntax quality of the augmented instances is an urgent problem. Therefore, BoostAug may not be an best choice for some tasks or models requiring syntax as an essential modeling objective (Zhang et al., 2019). In other words, the syntax quality of BoostAug depends on the backend.

References

Wei Bi, Huayang Li, and Jiacheng Huang. 2021. Data augmentation for text generation without any augmented data. In *ACL/IJCNLP'21: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 2223–2237. Association for Computational Linguistics.

Thomas Body, Xiaohui Tao, Yuefeng Li, Lin Li, and Ning Zhong. 2021. Using back-and-forth translation to create artificial augmented textual data for sentiment analysis models. *Expert Syst. Appl.*, 178:115033.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP'15: Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. The Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS'20: Advances in Neural Information Processing Systems*.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021. Neural data-to-text generation with lm-based text augmentation. In *EACL'21: Proc. of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 758–768. Association for Computational Linguistics.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL'20: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Comput. Speech Lang.*, 13(4):359–393.

Claude Coulombe. 2018. Text data augmentation made simple by leveraging NLP cloud apis. *CoRR*, abs/1812.04718.

Junqi Dai, Hang Yan, Tianxiang Sun, Pengfei Liu, and Xipeng Qiu. 2021. Does syntax matter? A strong baseline for aspect-based sentiment analysis with roberta. In *NAACL-HLT'21: Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1816–1829. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT'19: Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics.

Xin Dong, Yaxin Zhu, Zuohui Fu, Dongkuan Xu, and Gerard de Melo. 2021. Data augmentation with adversarial training for cross-lingual NLI. In *ACL/IJCNLP'21: Proc. of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5158–5167. Association for Computational Linguistics.

Ronald L. Graham. 1972. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Process. Lett.*, 1(4):132–133.

Giannis Haralabopoulos, Mercedes Torres Torres, Ioannis Anagnostopoulos, and Derek McAuley. 2021. Text data augmentations: Permutation, antonyms and negation. *Expert Syst. Appl.*, 177:114769.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *ICLR'21: 9th International Conference on Learning Representations*. OpenReview.net.

Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. 2019. A challenge dataset and effective models for aspect-based sentiment analysis. In *EMNLP-IJCNLP'19: Proc. of the 2019 Conference*

on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 6279–6284. Association for Computational Linguistics.

Ehsan Kamalloo, Mehdi Rezagholizadeh, and Ali Ghodsi. 2022. When chosen wisely, more data is what you need: A universal sample-efficient strategy for data augmentation. In *ACL'22: Findings of the Association for Computational Linguistics*, pages 1048–1062. Association for Computational Linguistics.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. AEDA: an easier data augmentation technique for text classification. In *EMNLP'21: Findings of the Association for Computational Linguistics*, pages 2748–2754. Association for Computational Linguistics.

Hazel H. Kim, Daecheol Woo, Seong Joon Oh, Jeong-Won Cha, and Yo-Sub Han. 2022. ALP: data augmentation using lexicalized pcfgs for few-shot text classification. In *AAAI'22: Thirty-Sixth AAAI Conference on Artificial Intelligence*, pages 10894–10902. AAAI Press.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha P. Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *NAACL-HLT'19: Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3609–3619. Association for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *CoRR*, abs/2003.02245.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL'20: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics.

Patrick S. H. Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. 2021. PAQ: 65 million probably-asked questions and what you can do with them. *Trans. Assoc. Comput. Linguistics*, 9:1098–1115.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *NDSS'19: 26th Annual Network and Distributed System Security Symposium*. The Internet Society.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Qiaoyang Luo, Lingqiao Liu, Yuhao Lin, and Wei Zhang. 2021. Don't miss the labels: Label-semantic augmented meta-learner for few-shot text classification. In *ACL/IJCNLP'21: Findings of the Association for Computational Linguistics*, volume ACL/IJCNLP 2021, pages 2773–2782. Association for Computational Linguistics.

Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In *SIGMOD'21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, pages 1303–1316. ACM.

Nikolaos Mittas and Lefteris Angelis. 2013. Ranking and clustering software cost estimation models through a multiple comparisons algorithm. *IEEE Trans. Software Eng.*, 39(4):537–551.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: self-supervised manifold based data augmentation for improving out-of-domain robustness. In *EMNLP'20: Proc. of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 1268–1283. Association for Computational Linguistics.

Tong Niu and Mohit Bansal. 2018. Adversarial over-sensitivity and over-stability strategies for dialogue models. In *CoNLL'18: Proc. of the 22nd Conference on Computational Natural Language Learning*, pages 486–496. Association for Computational Linguistics.

Minh Hieu Phan and Philip O. Ogunbona. 2020. Modelling context and syntactical features for aspect-based sentiment analysis. In *ACL'20: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3211–3220. Association for Computational Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia V. Loukachevitch, Evgeniy V. Kotelnikov, Núria Bel, Salud María Jiménez Zafra, and Gülsen Eryigit. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *NAACL-HLT'16: Proc. of the 10th International Workshop on Semantic Evaluation*, pages 19–30. The Association for Computer Linguistics.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *NAACL-HLT'15: Proc. of the 9th International Workshop on Semantic Evaluation*, pages 486–495. The Association for Computer Linguistics.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *ACL'14: Proc. of the 8th International Workshop on Semantic Evaluation*, pages 27–35. The Association for Computer Linguistics.

Shuhuai Ren, Jinchao Zhang, Lei Li, Xu Sun, and Jie Zhou. 2021. Text autoaugment: Learning compositional augmentation policy for text classification. In *EMNLP'21: Proc. of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9029–9043. Association for Computational Linguistics.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *EACL'12: 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549. The Association for Computer Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL'16: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021. Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning. In *ACL/IJCNLP'21: Findings of the Association for Computational Linguistics*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1569–1576. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP'13: Proc. of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. ACL.

Siyuan Wang, Wanjun Zhong, Duyu Tang, Zhongyu Wei, Zhihao Fan, Daxin Jiang, Ming Zhou, and Nan Duan. 2022a. Logic-driven context extension and data augmentation for logical reasoning of text. In *ACL'22: Findings of the Association for Computational Linguistics*, pages 1619–1629. Association for Computational Linguistics.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *EMNLP'16: Proc. of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615. The Association for Computational Linguistics.

Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. 2022b. Promda: Prompt-based data augmentation for low-resource NLU tasks. In *ACL'22: Proc. of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 4242–4255. Association for Computational Linguistics.

Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. 2022c. Promda: Prompt-based data augmentation for low-resource NLU tasks. In *ACL'22: Proc. of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 4242–4255. Association for Computational Linguistics.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP-IJCNLP'19: Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 6381–6387. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL'18: Proc. of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1112–1122. Association for Computational Linguistics.

Xing Wu, Chaochen Gao, Meng Lin, Liangjun Zang, and Songlin Hu. 2022. Text smoothing: Enhance various data augmentation methods on text classification tasks. In *ACL'22: Proc. of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 871–875. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *NeurIPS'20: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems*.

Kevin Yang, Olivia Deng, Charles Chen, Richard Shin, Subhro Roy, and Benjamin Van Durme. 2022. Addressing resource and privacy constraints in semantic parsing through data augmentation. In *ACL'22: Findings of the Association for Computational Linguistics*, pages 3685–3695. Association for Computational Linguistics.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woo-Myoung Park. 2021. Gpt3mix: Leveraging large-scale language models for text augmentation. In *EMNLP'21: Findings of the Association for Computational Linguistics*, pages 2225–2239. Association for Computational Linguistics.

Tianchi Yue, Shulin Liu, Huihui Cai, Tao Yang, Shengkang Song, and Tinghao Yu. 2022. Improving chinese grammatical error detection via data augmentation by conditional error generation. In *ACL'22:*

Findings of the Association for Computational Linguistics, pages 2966–2975. Association for Computational Linguistics.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *EMNLP-IJCNLP'19: Proc. of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4567–4577. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS'15: Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, pages 649–657.

Jing Zhou, Yanan Zheng, Jie Tang, Li Jian, and Zhilin Yang. 2022a. Flipda: Effective and robust data augmentation for few-shot learning. In *Proc. of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8646–8665. Association for Computational Linguistics.

Ran Zhou, Xin Li, Ruidan He, Lidong Bing, Erik Cambria, Luo Si, and Chunyan Miao. 2022b. MELM: data augmentation with masked entity language modeling for low-resource NER. In *ACL'22: Proc. of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 2251–2262. Association for Computational Linguistics.

A Hyperparameter Settings

A.1 Hyperparameter Settings for BoostAug

Some important parameters are set as follows.

- k is set to 5 for the k-fold cross-boosting on all datasets.
- The number of augmentation instances per example $\tilde{N}$ is 8.
- The transformation probability of each token in a sentence is set to 0.1 for all augmentation methods.
- The fixed confidence and perplexity thresholds are set as $\alpha = 0.99$ and $\beta = 5$ based on grid search. We provide sensitivity test of α and β in Appendix C.2.
- The learning rates of base models `LSTM` and `DeBERTa-BASE` are set as 10^{-3} and 10^{-5}, respectively.
- The batch size and maximum sequence modeling length are 16 and 80, respectively.
- The L_2 regularization parameter λ is 10^{-8}; we use Adam as the optimizer for all models during the training process.

B Baseline Backends

We use `BoostAug` to improve five state-of-the-art baseline text augmentation methods, all of which are used as the text augmentation back end of `BoostAug`. Please refer to Table 6 for detailed experimental results.

- `EDA`(TextAttack[8]) (Wei and Zou, 2019) performs text augmentation via random word insertions, substitutions and deletions.
- `SynonymAug`(NLPAug[9]) (Niu and Bansal, 2018) replaces words in the original text with their synonyms. This method has been shown to be effective in improving the robustness of models on certain tasks.
- `SpellingAug` (Coulombe, 2018): it substitutes words according to spelling mistake dictionary.
- `SplitAug` (Li et al., 2019) (NLPAug): it splits some words in the sentence into two words randomly.
- `BackTranslationAug` (Sennrich et al., 2016) (NLPAug): it is a sentence level augmentation method based on sequence translation.
- `ContextualWordEmbsAug` (Kumar et al., 2020) (NLPAug): it substitutes similar words ac-

[8] https://github.com/QData/TextAttack
[9] https://github.com/makcedward/nlpaug

cording to the PLM (i.e., `Roberta-base` (Liu et al., 2019)) given the context.

C Additional Experiments

C.1 Natural Language Inference Experiments

The experimental results in Table 5 show that the performance of both BERT and DeBERTa models can be improved by applying BoostAug. With BoostAug, the accuracy of the BERT model on SNLI improves from 70.72% to 73.08%, and on MNLI from 51.11% to 52.49%. The DeBERTa model also shows significant improvement with EDA, achieving 86.39% accuracy on SNLI and 78.04% on MNLI. These results demonstrate the effectiveness of BoostAug in improving the generalizability of natural language inference models, and its compatibility with different state-of-the-art pre-trained models such as BERT and DeBERTa.

Table 5: The additional experimental results on the SNLI and MNLI datasets for natural language inference. The back end of BoostAug is EDA.

Augmentation	Model	SNLI		MNLI	
		Acc	F1	Acc	F1
None	BERT	70.72	72.8	51.11	50.47
	DeBERTa	83.50	83.47	74.75	74.62
BoostAug	BERT	73.08	71.57	52.49	50.91
	DeBERTa	86.39	86.16	78.04	77.04

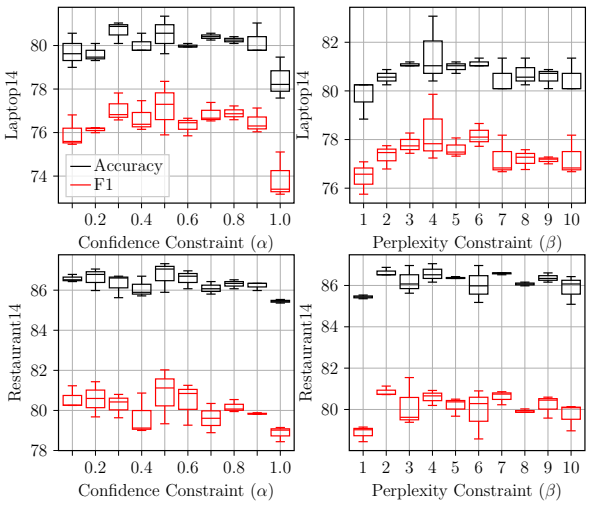

Figure 5: The performance box plots under different α and β in BoostAug(EDA).

C.2 Hyper-parameter Sensitivity Experiment

We provide the experimental results of BoostAug on the `Laptop14` and `Restaurant14` datasets in Figure 5.

C.3 Performance of BoostAug on Different Backends

To investigate the generalization ability of BoostAug, we evaluate its performance based on the existing augmentation backends. From the results shown in Table 6, we find that the performance of these text augmentation back ends can be improved by using our proposed BoostAug. Especially by cross-referencing the results shown in Table 2, we find that the conventional text augmentation methods can be enhanced if appropriate instance filtering strategies are applied.

Another interesting observation is that PLMs are not effective for text augmentation, e.g., `WordEmdsAug` is outperformed by `EDA` in most comparisons[10]. Moreover, PLMs are resource-intense and usually cause a biased feature space. This is because PLMs can generate some unknown words, which are outside the testing set, during the pre-training stage. Our experiments indicate that using PLM as an augmentation instance filter, instead of a text augmentation tool directly, can help alleviate the feature space shift.

C.4 Visualization of feature space

Figure 6 shows the feature space shift of the ABSC datasets, where the augmentation back end of BoostAug is EDA.

C.5 Trajectory Visualization of RQ4

Figure 7 shows the performance trajectory visualization of `MonoAug` and `EDA`. Compared to BoostAug, `MonoAug` and existing augmentation methods usually trigger performance sacrifice while augmentation instances for each example are more than 3.

[10] In fact, we also tried some other PLM-based augmentation back ends, e.g., `BackTranslationAug`, and we come up with same observation.

Table 6: Performance comparison of BOOSTAUG based on different augment back ends.

Backend	Model	MAMS		SST2		SST5		AGNews10K	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
None	LSTM	56.96	56.18	82.37	82.37	44.39	43.60	87.60	87.36
	BERT	82.78	82.04	90.77	90.76	52.90	53.02	92.47	92.26
	DeBERTa	83.31	82.87	95.28	95.28	56.47	55.58	92.30	92.13
EDA	LSTM	59.21	59.58	85.83	85.83	45.93	43.59	88.45	88.16
	BERT	84.01	83.44	92.33	92.33	53.94	52.80	92.48	92.25
	DeBERTa	**84.51**	**83.97**	**96.09**	**96.09**	57.78	**56.15**	92.95	92.76
SpellingAug	LSTM	58.50	57.65	85.23	85.23	43.39	42.45	87.93	87.63
	BERT	83.23	82.70	92.01	92.01	52.26	51.03	91.82	91.59
	DeBERTa	83.98	83.44	95.22	95.22	**57.91**	55.88	92.77	92.54
SplitAug	LSTM	58.65	57.23	85.64	85.64	46.04	43.97	87.65	87.42
	BERT	83.05	82.49	92.20	92.20	51.86	51.39	91.92	91.69
	DeBERTa	82.67	82.26	94.76	94.76	57.67	55.90	92.70	92.51
WordEmdsAug	LSTM	59.54	57.58	86.30	86.30	46.47	44.15	88.38	88.10
	BERT	83.31	82.72	91.76	91.76	52.49	50.27	92.43	92.24
	DeBERTa	83.35	82.87	95.33	95.33	57.22	56.08	**93.88**	**93.70**

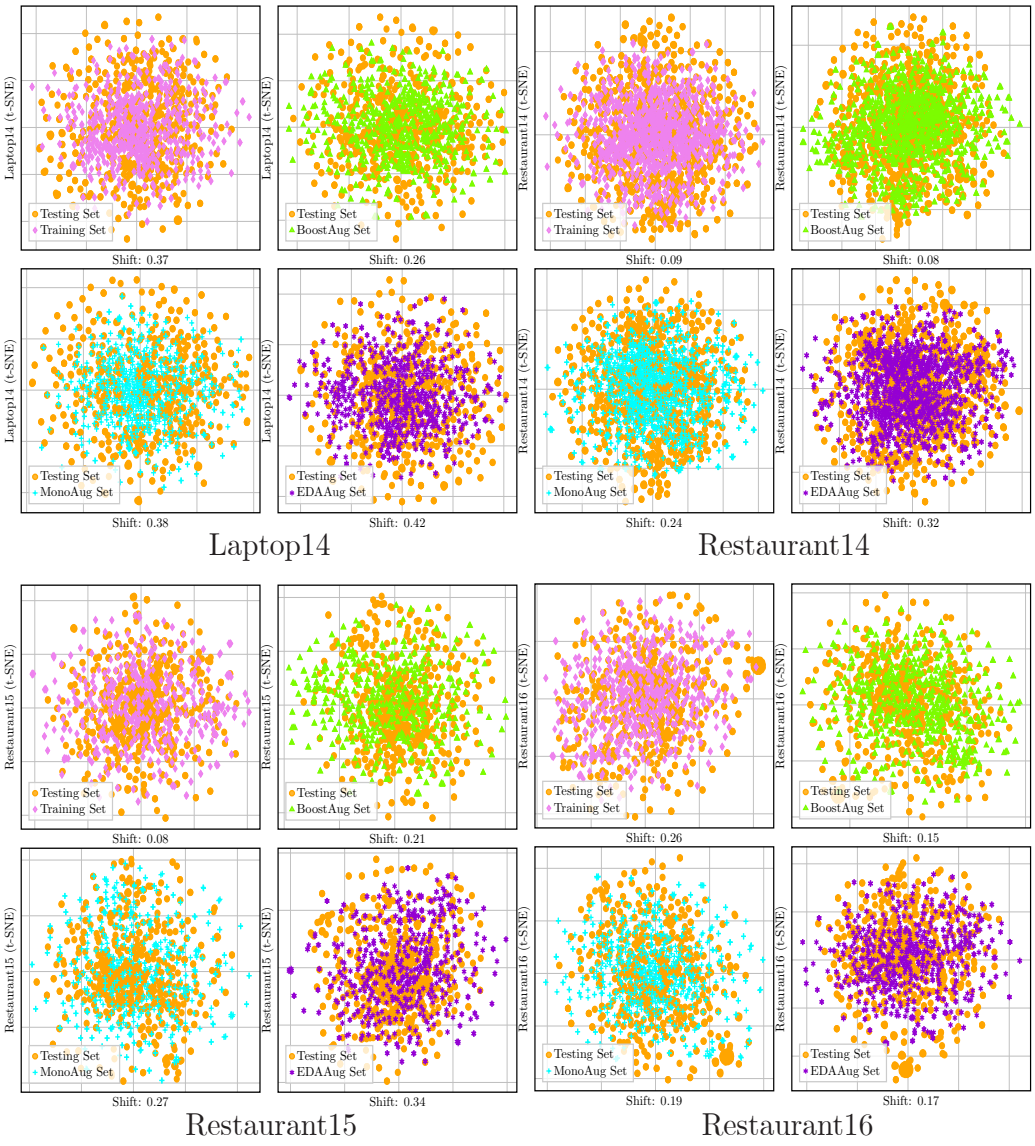

Figure 6: This figure shows the feature space shift ($\mathcal{S}$) of four ABSC datasets as visualized by t-SNE. The results demonstrate that BOOSTAUG has the least feature space shifts in comparison to other augmentation methods, such as MONOAUG and EDA.

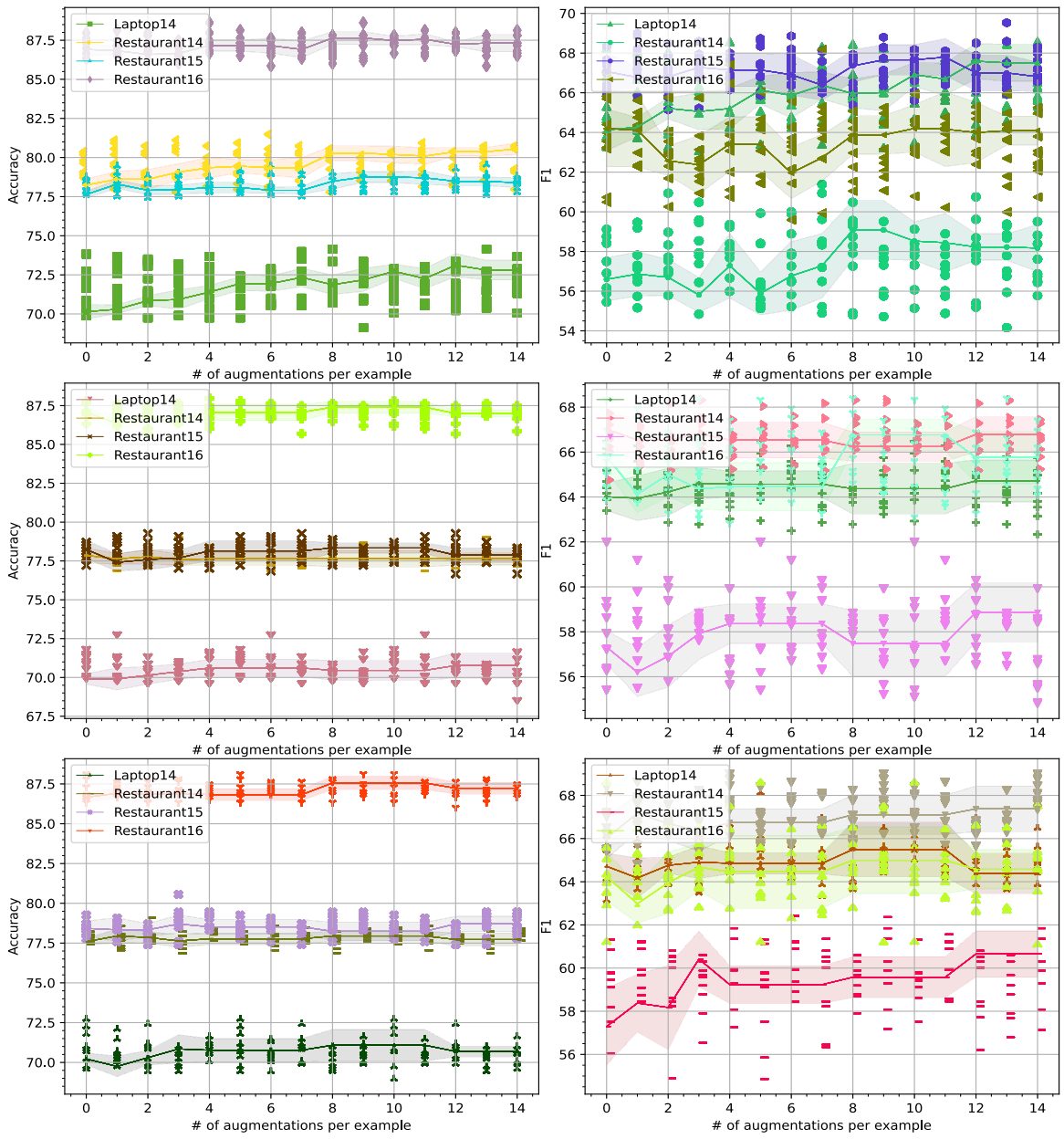

Figure 7: The performance (i.e., classification accuracy and F1 score) visualization of how BOOSTAUG perform as the number of augmentation instances per example increases.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
7

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

3

☑ B1. Did you cite the creators of artifacts you used?
3

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
The licenses of all artifacts are well announced on their websites

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
3

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. The documentation of all artifacts are well organized on their websites

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
3

C ☑ Did you run computational experiments?

4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix B.1

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.