Augmentor or Filter? Reconsider the Role of Pre-trained Language Model in Text Classification Augmentation

Anonymous ACL submission

Abstract

Text augmentation is one of the most effective techniques to solve the critical problem of insufficient data in text classification. Existing text augmentation methods achieve hopeful performance in few-shot text data augmentation. However, these methods usually lead to performance degeneration on public datasets due to poor quality augmentation instances. Our study shows that even employing pre-trained language models, existing text augmentation methods generate numerous low-quality instances and lead to the feature space shift problem in augmentation instances. However, we note that the pre-trained language model is good at finding low-quality instances provided that it has been fine-tuned on the target dataset. To alleviate the feature space shift and performance degeneration in existing text augmentation methods, we propose BoostAug, which reconsiders the role of the language model in text augmentation and emphasizes the augmentation instance filtering rather than generation. We evaluate BoostAug on both sentence-level text classification and aspect-based sentiment classification. The experimental results on seven commonly used text classification datasets show that our augmentation method obtains state-of-the-art performance. Moreover, BoostAug is a flexible framework; we release the code which can help improve existing augmentation methods.

1 Introduction

Due to the progress of pre-training techniques in natural language processing, the pretrained language models (Devlin et al., 2019; He et al., 2021)(PLM) has been capable of learning large-scale data. As a result, data insufficiency has been an urgent problem in many low-resource NLP tasks. To mitigate the above problem, many text augmentation works (Sennrich et al., 2016; Coulombe, 2018; Li et al., 2019; Wei and Zou, 2019; Kumar et al., 2020) that aim at improving text classification have been proposed in recent studies. These works achieve hopeful improvement in few-shot augmentation experiments, while most of them encounter a failure mode (Zhou et al., 2021) in relatively large public datasets. Recent works (Body et al., 2021; Chang et al., 2021; Luo et al., 2021) expect to leverage the language modeling ability of PLMs in text augmentation, but these methods can degenerate the performance as well.

To explore the crux of the augmentation failure mode, we conduct experiments to prove that the existing text augmentation methods generate plenty of unnecessary augmentation instances. These low-quality augmentation cause the feature space shift problem and degrade the performance. Unfortunately, these unnecessary augmentation instances can be introduced in both text edit-based and text generation-based augmentation methods. For the edit-based methods, the low-quality instances mainly come from some breaking text transformation (e.g., changing a word 'greatest' to 'worst' in a sentence leads to an adverse meaning in a sentiment analysis task.), while generation-based methods usually introduce out-of-distribution words due to synonym replacement and word insertion. To solve the performance degeneration of existing augmentation methods, we intend to alleviate the feature space shift caused by low-quality instances. According to our preliminary research, we notice that the PLMs fine-tuned on the targeted dataset will be familiar with the identical distribution data. In other words, the fine-tuned PLMs always have high confidence and lower perplexity for identical distribution text. Motivated by this finding, we reconsider the role of PLMs in text augmentation and propose an instance-filter framework, BoostAug, based on DeBERTa (He et al., 2021). BoostAug adopts the existing text augmentation methods (Wei and Zou, 2019; Coulombe, 2018; Li et al., 2019; Kumar et al., 2019) as backends and consists of three instance filtering strate-

gies(which will be discussed in Section 2.3): perplexity filtering, confidence ranking and predicted label constraint. Moreover, we introduce a cross-boosting strategy for BOOSTAUG to alleviate biased filter training in Section 2.1. According to our experimental results on four aspect-based sentiment classification and two sentence-level text classification datasets, we find BOOSTAUG significantly alleviates the feature space shift problem in existing augmentation methods. Moreover, due to the mitigation of feature space shift, BOOSTAUG can generate more augmentation instances and improve the performance; on the contrary, more augmentation instances trigger performance degeneration in other methods.

To the best of our knowledge, BOOSTAUG is the first text augmentation method that emphasizes the instance-quality control and applies the PLMs as filters instead of augmentor. The experiment results[1] show that BOOSTAUG outperforms existing text augmentation methods and achieves state-of-the-art performance on ABSC and TC tasks. Therefore, our main contribution concludes:

- We explore the crux for performance degeneration in existing text augmentation methods and emphasize the importance of augmentation instance quality control.
- We propose a PLM-based augmentation instance filter BOOSTAUG to mitigate feature space shift and significantly improve the performance on the ABSC and TC tasks according to the experimental results.
- Our experiments show that the existing text augmentation methods can be improved by employing an instance filter, indicating BOOSTAUG is available for improving existing text augmentation methods.

2 Proposed Method

The workflow of our proposed BOOSTAUG for text classification tasks is shown in Figure 1 and the pseudo code is given in Algorithm 1. Different from most existing studies, which focus on an unsupervised instance generation, BOOSTAUG mainly works on the selection of high-quality augmentation instances. Generally speaking, it consists of two major phases: one is to build a surrogate language model; while the other is to use the surrogate language model as the driver to guide the augmentation instance generation. In the following paragraphs, we will delineate their implementations step by step.

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 $\quad \mathcal{D}^i_{\text{aug}} := \emptyset$, $\mathcal{D}^i_{\text{boost}} := \mathcal{F}^i$;
5 $\quad$ Randomly pick up $k-2$ folds except $\mathcal{F}^i$ to constitute $\mathcal{D}^i_{\text{train}}$;
6 $\quad \mathcal{D}^i_{\text{valid}} := \mathcal{F} \setminus (\mathcal{F}^i \bigcup \mathcal{D}^i_{\text{train}})$;
7 $\quad$ Use the DeBERTa on $\mathcal{D}^i_{\text{train}}$ and $\mathcal{D}^i_{\text{valid}}$ to build the surrogate language model;
8 $\quad$ **forall** $d_{\text{org}} \in \mathcal{D}^i_{\text{boost}}$ **do**
9 $\quad\quad \mathcal{D}^i_{\text{aug}} := F(d^i_{\text{org}}, \tilde{N}, \Theta)$;
10 $\quad\quad$ **forall** $d_{\text{aug}} \in \mathcal{D}^i_{\text{aug}}$ **do**
11 $\quad\quad\quad$ 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 $\quad\quad\quad$ **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 $\quad\quad\quad\quad \mathcal{D}^i_{\text{aug}} := \mathcal{D}^i_{\text{aug}} \setminus \{d_{\text{aug}}\}$;
14 $\quad\quad \mathcal{D}_{\text{aug}} := \mathcal{D}_{\text{aug}} \bigcup \mathcal{D}^i_{\text{aug}}$;
15 $\quad \mathcal{D}_{\text{aug}} := \mathcal{D}_{\text{aug}} \bigcup \mathcal{D}^i_{\text{boost}}$;
16 **return** $\mathcal{D}_{\text{aug}}$

2.1 Building a surrogate language model

At the beginning of the Phase #1, the original training dataset is divided into $k > 3$ folds where the $k-2$ ones are used for the training purpose (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 identical to the data used for training the surrogate language model. This is called a data overlapping problem that leads to a feature space shift thus overfit the surrogate language model. 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 the Phase #1 is to build a surrogate language model as a filter

[1] We release the source code and experiment scripts of BOOSTAUG in the supplementary materials to help reproduce the experimental results.

[2] Note that we iteratively select the i-th fold, $i \in \{1, \cdots, k\}$, as the boosting fold (line 3 in Algorithm 1). Meanwhile, we randomly pick up the other $k-2$ folds to constitute the training folds and the remaining one as the validation fold. In particular, the validation fold is used to select the best checkpoint of the surrogate language model.

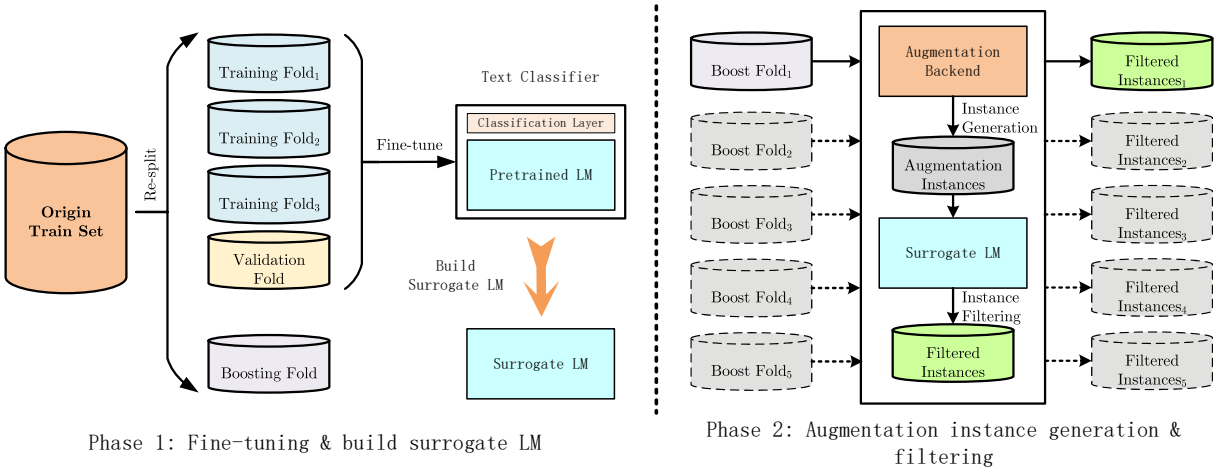

Figure 1: The workflow of BoostAug. In Phase #1, we fine-tune the DeBERTa-based classification model using the re-split training and validation sets and extract the fine-tuned DeBERTa to build a surrogate language model. In Phase #2, BoostAug employs a text augmentation backend to generate raw augmentations and filter the low-quality instance identified by the surrogate language model. BoostAug performs k-fold cross-boosting to avoid boosting, so Phase #1 and Phase #2 repeat k times.

to guide the elimination of harmful and poor augmentation instances. This is different from the 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 such as the text perplexity, classification confidence and predicted labels that can be useful for the quality control of the augmentation instances.
- Comparing to the instance generation, we argue that the instance filtering approach can be readily integrated with any existing text augmentation approach.

In practice, we first build a temporary classification model based on DeBERTa (He et al., 2021). Thereafter, it is fine-tuned based on the data in the $k-2$ training folds and the validation fold to learn the semantic features therein (line 7). Note that we do not use the original training dataset to carry out this fine-tuning. At the end, the language model built from the DeBERTa classification model is thus used as the surrogate language model for the instance filtering step in the Phase #2 of BoostAug.

2.2 Augmentation instance generation

As a building block of the 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. At 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 strategies to constitute the transformation function. For example, the EDA developed in (Wei and Zou, 2019) has four transformation strategies including synonym replacement, random insertion, random swap, 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 the 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 the perplexity filtering, the confidence ranking, and the predicted label constraint which will be delineated in the following paragraphs, respectively. Note that all these filtering strategies are built up the surrogate language model developed in the 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 the low-quality instances have a relatively high perplexity. This indicates that the perplexity information can be used to evaluate the quality of an augmentation instance. Since the surrogate language model built in the Phase #1 is bidirectional, the text perplexity of an augmentation instance d_{aug} is calculated as:

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

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 an low-quality instance and is dumped if $\mathbb{P}(d_{\text{aug}}) \geq \alpha$ while $\alpha \geq 0$ is a predefined threshold.

2.3.2 Confidence ranking

Our preliminary experiments have shown that the fine-tuned surrogate language model is always confident about the data in the training folds (or those having the identical distribution) where the confidence level is larger than 99%. However, we observe a significant feature space shift in the augmentation instances when there exist fatal errors in the grammars, syntaxes and words of the augmentation instances generated by the augmentation back ends. These instances will be allocated with a low confidence 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. Note that since the classification confidence of most augmentation instances are larger than 95%, it can lead to an unbalanced distribution if we simply use the classification confidence as the criterion for instance filtering. In particular, such unbalanced distribution can be attributed to the following two aspects.

- The surrogate language model is difficult to capture the change of several words for the long texts whereas it can be sensitive to deleting or adding words for the short texts.
- There can be various transformations leading to valid augmentation instances whereas the alternatives for the short texts are limited.

Therefore, it is natural that the long texts can have way more augmentation instances than the short texts, thus leading to the so called unbalanced distribution. In BOOSTAUG, to mitigate the side effect brought by the unbalanced distribution, we develop a confidence ranking strategy to eliminate the redundant augmentation instances generated from the long texts while retain 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:

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

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

2.3.3 Predicted label constraint

Due to some breaking text transformation, the 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.

Table 1: The split of different datasets.

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

3 Experimental Setup

3.1 Datasets

Our experiments are conducted on two classification tasks including the sentence-level text classification (TC) and the aspect-based sentiment classification (ABSC). SST-2, SST-5 (Socher et al., 2013) from the Stanford Sentiment Treebank and AGNews10K[3] (Zhang et al., 2015)are used to constitute the datasets for the TC task; while the datasets 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 split of these datasets are summarized in Table 1. The widely used accuracy (Acc) and macro F1 as used as the metric for evaluating the performance of different algorithms following existing research (Wang et al., 2016; Zhou et al., 2021). In particular, all experiments are repeated five times with a different random seed.

3.2 Augment Backends

The performance of BoostAug is compared with four state-of-the-art (SOTA) word-level text augmentation methods, all of which are used as the text augmentation back end of BoostAug.

- EDA (Wei and Zou, 2019) (TextAttack[4]): it performs augmentation via random word insertions, substitutions and deletions.
- SpellingAug (Coulombe, 2018) (NLPAug[5]): 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.
- WordEmbsAug (Kumar et al., 2020) (NLPAug): it substitutes similar words according to the PLM, Roberta-base (Liu et al., 2019) in particular, given the context.

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

3.3 Parameter Settings

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 $\beta = 5$.
- The learning rates of 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.

4 Experimental Results

4.1 Comparison with Four Selected Text Augmentation Methods

From the comparison results shown in Table 2, it is clear to see that the overall performance improvement of using BoostAug is consistently better than the other four peer text augmentation methods. It is also worth noting that the performance of some classification models can be degraded by using a conventional text augmentation methods. Another interesting observation is the performance improvement is relatively marginal for a large dataset like SST2, SST5 and MAMS. Furthermore, the performance of LSTM is more sensitive to the data augmentation because it lacks knowledge learned from a large-scale corpus compared to the PLMs.

As for the selected peer text augmentation methods, the best one is EDA, even though it is the simplest, given that its augmentation strategies are more resilient to the distribution of the feature space. In contrast, SplitAug is almost the

[3]We collect first 10K examples to build the AGNews10K dataset ($7K$ for training set, $1K$ for validation set and $2K$ for testing), and it is large enough compared to other datasets.

[4]https://github.com/QData/TextAttack

[5]https://github.com/makcedward/nlpaug

Table 2: Performance comparison of BoostAug w.r.t. four selected text augmentation methods. The best metric values are highlighted in **bold** face.

Method	Model	Laptop		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	—	—	—	—	—	—
WordEmbsAug	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	—	—	—	—	—	—
BoostAug	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†	—	—	—	—	—	—

None indicates the vanilla version without using a text augmentation. † indicates the best method is significantly better than the underlying peeri according to the Wilcoxon's rank sum test at a 0.05 significance level.

Table 3: Performance comparison of different ablation variants of BoostAug.

Variants	Model	MAMS		SST2		SST5		AGNews	
		Acc	F1	Acc	F1	Acc	F1	Acc	F1
BoostAug	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**	**57.52**	**56.48**	**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	57.1	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	58.1	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

worst for LSTM because its augmentation instances suffer from a severe feature space bias due to its word split transformation. The performance of SpellingAug is comparable with EDA. This can be understood as the PLMs capture some frequent misspellings during the pre-training process. Note that the PLM-based augmentation methods, including WordsEmbsAug, tend to generate instances with unknown words for the original data. This further aggregates the feature space shift of the augmented data.

4.2 Ablation study for BoostAug

To better understand the working mechanism of BoostAug, we develop a couple of variants to evaluate the effectiveness of the cross-boosting, the predicted label constraint, the confidence ranking and the perplexity filtering in this subsection. As the results shown in Table 3, it is clear to see a performance degradation of MonoAug which takes out the cross-boosting of BoostAug. Since the entire training set is used to train a surrogate language model in MonoAug, the augmentation instances are thus very similar to the original data. This is the data overlapping problem discussed in Section 2.1 that leads to a biased instance filtering and makes the filtered instances overfit the data distribution of training folds. The variant without using the perplexity filtering strategy is the worst as shown in Table 3. This is because the perplexity filtering strategy can help remove the instances with syntactical and grammar errors. The performance of the variants without using the predicted label constraint and the confidence ranking is similar. In particular, without using the label constraint, the features of the augmented samples can always mutate to an adverse meaning. As for the confidence ranking, it helps eliminate many out of domain words that lead to the feature space shift.

4.3 Investigation of the feature space shift

This subsection empirically investigates the feature space shift problem mentioned in Section 2.1. To this end, we apply t-SNE (Hinton and Roweis, 2002) to visualize the distribution of the features of the testing set versus different augmented variants. Furthermore, we evaluate the convex hull (Graham, 1972) overlapping rate and the distribution skewness (Kokoska and Zwillinger, 2000) between the

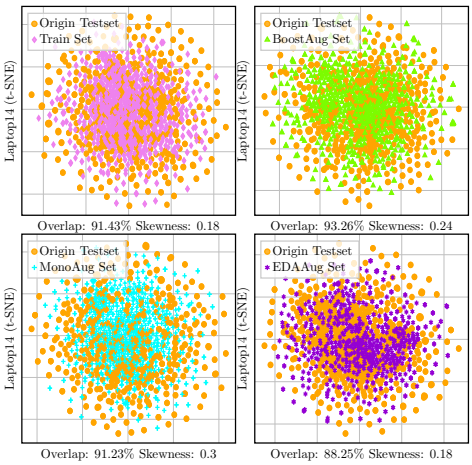

Figure 2: The t-SNE visualization of feature space of the Laptop14 dataset.

augmentation instances and the testing set[6]. In particular, the larger the overlapping rate is (or the smaller the skewness is), the smaller feature space shift achieved. The t-SNE plots of Laptop14 are given in Figure 2 while the full results are available in Figure 6 of the appendix. From the experiment results, it is clear to see that the augmentation instances generated by BOOSTAUG have the mildest feature space shift. In particular, the overlapping rate and the skewness w.r.t. the testing set are consistently better than the training set. This explains the performance improvement achieved by using BOOSTAUG as discussed in Section 4.1. In contrast, the augmentation instances generated by EDA, the best peer text augmentation methods as observed in Section 4.1, have a worse overlapping rate even compared to the training set. This explains the performance degradation when using EDA on the baseline classification models observed in Section 4.1. It is also interesting to note that the quality of the augmentation instances generated by MonoAug is better than EDA.

4.4 Comparison of using BOOSTAUG on different back ends

To investigate the generalization ability of BOOSTAUG, we evaluate its performance based on the existing augmentation backends. From the results shown in Table 4, 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.

[6]The implementations of overlapping rate and skewness calcualation are introduced in Section A.

Table 4: Performance comparison of BOOSTAUG based on different augment back ends.

Augment Method	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	58.83	57.42	85.77	85.77	45.25	43.84	88.45	88.16
	BERT	83.38	82.97	92.33	92.33	51.21	50.09	92.48	92.25
	DeBERTa	83.68	83.00	**96.09**	**96.09**	**58.51**	**57.06**	92.88	92.70
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**

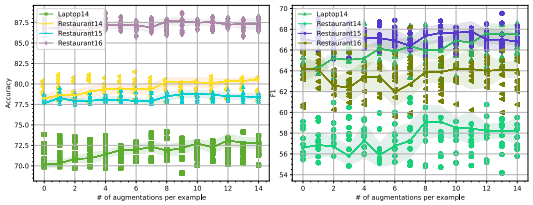

Figure 3: Trajectories of the accuracy and the F1 values with error bars versus the number of augmentation instances generated for an example by using BOOSTAUG.

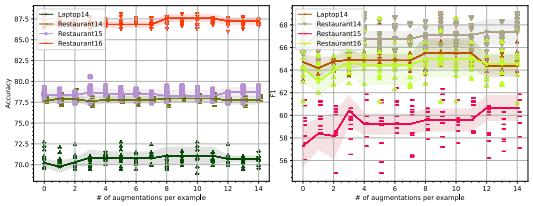

Figure 4: The visualization of MonoAug's performance based on different numbers of augmentation per example.

Another interesting observation is that PLMs are not effective for text augmentation, e.g., WordEmdsAug is outperformed by EDA in most comparisons[7]. 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.

4.5 How many augmentation instances do we actually need?

Although BOOSTAUG alleviate the feature space shift compared to existing text augmentation methods, we find that merely increasing the augmenta-

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

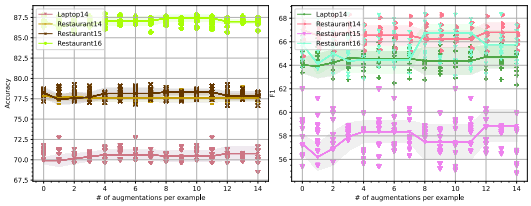

Figure 5: The visualization of EDA's performance based on different numbers of augmentation per example.

tion instances will ultimately amplify the feature space shift problem. In this subsection, we plan to empirically investigate the number of augmentation instances required for an original example under different circumstances. Figure 3 to 5 plot the trajectories of the accuracy and F1 with error bars versus the number of augmentation instances generated for each example by using BOOSTAUG, MONOAUG and EDA.

From these plots, we find that the performance of EDA and MONOAUG can be improved with the increase of the number of augmentation instances until $\tilde{N} \geq 3$. This can be attributed to the feature space shift caused by a deviation from the ground truth data distribution or a biased instance filtering. As for BOOSTAUG, we can see that generating more augmentation instances can help improve the performance (until $\tilde{N} \geq 8$). In particular, the F1 metric obtained by BOOSTAUG is consistently better than that of EDA and MONOAUG. It is also interesting to note that text augmentation is not always helpful, given that the performance can be better in case no text augmentation is involved when $\tilde{N} = 1$.

5 Related Works

Text augmentation has attracted more and more researchers. Although recent works on text augmentation primarily focus on the generation-only augmentation methods (Bi et al., 2021; Ren et al., 2021; Chang et al., 2021; Haralabopoulos et al., 2021; Body et al., 2021), the significance of augmentation instance filtering has been underestimated. Our work reformulates the text augmentation as an augment-to-filter problem, and BOOSTAUG accepts varieties of augmentation generation backends, e.g., EDA (Wei and Zou, 2019), SPELLINGAUG (Coulombe, 2018; Li et al., 2019). Apart from these generation-only methods, recent works (Sennrich et al., 2016; Kumar et al., 2020) recognize the significance of PLM for text augmentation, but their performance is limited on non-small datasets. Our work leverages the PLM as a non-exclusive filter that can improve generation-only methods. We apply crossing augmentation to mitigate overfitting problems while training the surrogate language model, and there is rare research working on augmentation quality measurement. We note the Lim et al. (2019) adopts crossing augmentation in image classification. However, it is used for improving augmentation efficiency instead of keeping feature space. Our work indicates that augmentation quality is more significant than quantity for prompting text classification.

6 Conclusion

We find existing text augmentation methods usually lead to performance degeneration in public datasets due to numerous low-quality augmentation instances. Hence, we reformulate the text augmentation problem as an augment-to-filter problem and propose BOOSTAUG to promote text classification tasks. Differing from existing text augmentation studies, BOOSTAUG emphasizes the importance of augmentation instance quality control. To eliminate low-quality augmentation instances, we apply a PLM-based surrogate language model as the filter to discriminate unnecessary instances according to perplexity filtering, confidence ranking, and predicted label constraint. Experimental results on three text classification datasets and five aspect-based sentiment classification datasets show that BOOSTAUG reclaim the state-of-the-art performance. Our research proves that augmentation instance filtering is as important as instance generation.

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 qual-

ity of the augmented instances is an urgent problem. Therefore, BOOSTAUG may not be an optimal choice for some tasks or models requiring syntax as an essential modeling objective (Zhang et al., 2019). In addition, once a text augmentation generation method considers syntax quality control, BOOSTAUG would be useful in these syntax-sensitive tasks.

References

Wei Bi, Huayang Li, and Jiacheng Huang. 2021. Data augmentation for text generation without any augmented data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, 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.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021. Neural data-to-text generation with lm-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 758–768. 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 *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, 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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. 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 *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Geoffrey E. Hinton and Sam T. Roweis. 2002. Stochastic neighbor embedding. In *NIPS'02: Proc. of Advances in Neural Information Processing Systems 15*, pages 833–840. MIT Press.

Qingnan Jiang, Lei Chen, Ruifeng Xu, Xiang Ao, and Min Yang. 2019. A challenge dataset and effective models for aspect-based sentiment analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6279–6284. Association for Computational Linguistics.

Stephen Kokoska and Daniel Zwillinger. 2000. *CRC standard probability and statistics tables and formulae*. CRC Press. Section 2.2.24.1.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha P. Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 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.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. 2019. Fast autoaugment. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 6662–6672.

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 *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 2773–2782. Association for Computational Linguistics.

Minh Hieu Phan and Philip O. Ogunbona. 2020. Modelling context and syntactical features for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, 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 *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, 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 *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, 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 *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, 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 *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9029–9043. Association for Computational Linguistics.

Rico Sennrich. 2012. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, 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 *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer 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 *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 606–615. The Association for Computational Linguistics.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6381–6387. Association for Computational Linguistics.

Chen Zhang, Qiuchi Li, and Dawei Song. 2019. Aspect-based sentiment classification with aspect-specific graph convolutional networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4567–4577. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Jing Zhou, Yanan Zheng, Jie Tang, Jian Li, and Zhilin Yang. 2021. Flipda: Effective and robust data augmentation for few-shot learning. *CoRR*, abs/2108.06332.

A Metrics for feature space shift

A.1 Convex hull overlapping calculation

To calculate the convex hull overlapping rate, we use the Graham Scan algorithm[8] (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:

$$\mathcal{O} = \frac{\mathcal{P}_1 \cap \mathcal{P}_2}{\mathcal{P}_1 \cup \mathcal{P}_2}, \tag{4}$$

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

A.2 Distribution skewness calculation

The skewness of a example distribution is computed as following:

$$sk = \frac{m_3}{m_2^{3/2}}, \tag{5}$$

$$m_i = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^i, \tag{6}$$

where N is the number of instance 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 have 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:

$$sk^g = |sk^x| + |sk^y|, \tag{7}$$

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.

B Visualization of feature space

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

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

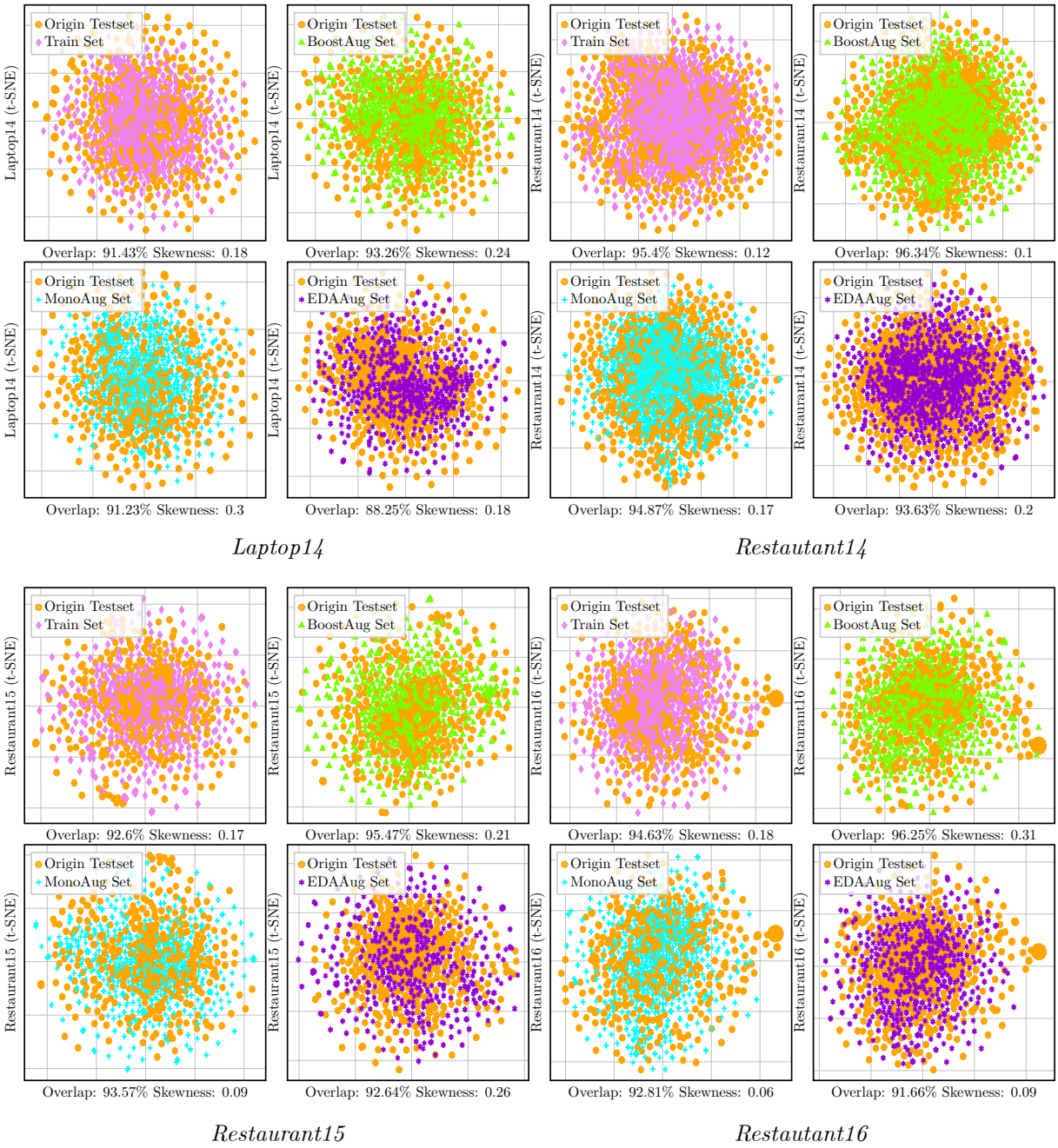

Figure 6: The t-SNE visualization of feature space of the ABSC datasets.