STA: Self-controlled Text Augmentation for Improving Text Classifications

Removed for anonymous review

Abstract

Despite recent advancements in Machine Learning, many tasks still involve working in low-data regimes which can make solving natural language problems difficult. Recently, a number of text augmentation techniques have emerged in the field of *Natural Language Processing* (NLP) which can enrich the training data with new examples, though they are not without their caveats. For instance, simple rule-based heuristic methods are effective, but lack variation in semantic content and syntactic structure with respect to the original text. On the other hand, more complex deep learning approaches can cause extreme shifts in the intrinsic meaning of the text and introduce unwanted noise into the training data. To more reliably control the quality of the augmented examples, we introduce a state-of-the-art approach for *Self-Controlled Text Augmentation* (STA). Our approach tightly controls the generation process by introducing a self-checking procedure to ensure that generated examples retain the semantic content of the original text. Experimental results on multiple benchmarking datasets demonstrate that *STA* substantially outperforms existing state-of-the-art techniques, whilst qualitative analysis reveals that the generated examples are both lexically diverse and semantically reliable.

1 Introduction

A variety of tasks such as *Topic Classification* (Li and Roth, 2002), *Emotion Detection* (Saravia et al., 2018) and *Sentiment Analysis* (Socher et al., 2013) have become important areas of research in NLP. Such tasks generally require a considerable amount of accurately labelled data to achieve strong performance. However, acquiring enough such data is both costly and time-consuming, hence making it rare in practice. This has motivated a vast body of research in techniques that can help alleviate issues associated with low-data regimes.

A popular augmentation approach involves the use of rule-based transformations, which employ intuitive heuristics based on well-known paradigmatic relationships between words. For instance, by using a lexical-semantic database such as *WordNet* (Miller, 1995), researchers can make rational and domain-specific conjectures about suitable replacements for words from lists of known synonyms or hyponyms/hypernyms (Wang and Yang, 2015; Wei and Zou, 2019; Feng et al., 2020). Whilst these substitution-based approaches can result in novel and lexically diverse data, they also tend to produce highly homogeneous structures, even when context-free grammars are used to generate more syntactically variable examples (Jia and Liang, 2016).

The recent success of pretrained transformer language models such as BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) has helped facilitate more robust strategies for dealing with low-resource scenarios: Conditional text generation. Large language models — typically trained on a vast corpus of text — contain a rich understanding of syntactic structure and semantic phenomena and thus can be well suited for faithful domain-specific generation (Petroni et al., 2019). Indeed, large language models have been conditioned to great success (Kobayashi, 2018; Wu et al., 2019; Anaby-Tavor et al., 2020; Kumar et al., 2020) to synthesize highly diverse training examples resulting in stronger downstream performance in low-resource settings. However, the use of diverse neurally-generated data may come at the cost of introducing semantic discrepancies, which can cause misalignment between the generated samples and their intended labels. Ideally, the optimal augmentation method would be one that satisfies both **Lexical/Syntactic Diversity** and **Semantic Fidelity** (reliable alignment between semantic meaning and class label).

In this paper, we propose a novel strategy — self-controlled text augmentation (STA) that aims to tightly control the generation process in order to produce diverse training examples which retain a high level of semantic fidelity. Following previous work, we fine-tune a state-of-the-art sequence-to-sequence transformer model, known as *T5* (Raffel et al., 2020), using a dataset containing only a limited number of samples and generate new samples using task-specific prompting, which has been shown to be effective in low-resource scenarios (Le Scao and Rush, 2021). While similar approaches have been deployed in previous work (Anaby-Tavor et al., 2020), our novel strategy effectively utilizes *Pattern-Exploiting Training* (Schick and Schütze, 2021a,b) by employing templates of verbalization-patterns that simultaneously direct the generation process and filter noisy labels within a single unified framework. Experimental results on multiple benchmarks demonstrate that STA outperforms existing state-of-the-art augmentation techniques. Furthermore, examining the quality of the augmented data reveals better diversity and fidelity as compared to the existing techniques.

2 Related Work

Various text augmentation techniques have been proposed in the literature. Zhang et al. (2015) and Wei and Zou (2019) use simple operations like synonym replacement, random insertion, swap, and deletion to generate new samples. Feng et al. (2020) further explores these substitution techniques for text generation. In contrast, Wang and Yang (2015) and Kobayashi (2018) use word embeddings and contextual language models, respectively, to replace words or phrases with semantically similar concepts.

Back translation is another effective method for text augmentation, transforming sentence between languages (Sennrich et al., 2016; Shleifer, 2019). Recently, researchers have explored the use of pre-trained transformer-based language models for conditional text augmentation to generate novel sentences from the original data (Wu et al., 2019; Anaby-Tavor et al., 2020; Kumar et al., 2020). For instance, Wu et al. (2019) leveraged BERT's masked language model, while Anaby-Tavor et al. (2020) fine-tuned GPT-2 to generate novel sentences and filter out noisy ones using a jointly trained classifier with some success in tackling the label misalignment problem. Similarly, Kumar et al. (2020) studied conditional text augmentation using transformer-based models, with BART outperforming other methods in low-resource settings

Building upon ideas presented in the GPT series (Radford et al., 2018, 2019; Brown et al., 2020), prompt-based templates have become and effective approach for eliciting latent knowledge from language models to great success (Trinh and Le, 2018; Petroni et al., 2019; Davison et al., 2019; Talmor et al., 2020; Le Scao and Rush, 2021). Wang et al. (2021) proposed using GPT-3 for text augmentation with zero-label learning, with results that were competitive when compared to fully supervised approaches. More closely related to our instruction-based generation strategy, Schick and Schütze (2021b) propose GenPet which is used to directly tackle a number of text generation tasks rather than text augmentation itself. In their work, which builds upon previous research PET (Schick and Schütze, 2021a), the authors alter the text inputs to form cloze-style questions known as prompting training (Liu et al., 2021), demonstrating improved performance on few-shot downstream tasks. Finally, researchers have proposed an array of techniques aiming to systematically engineer the structure of these templates beyond ad hoc human intuitive reasoning: For example, using automated template generation for the tasks (Shin et al., 2020; Gao et al., 2021), trained end-to-end with soft-prompts (Lester et al., 2021; Gu et al., 2022) or designed from sub-prompts created by decomposing prior task knowledge into rules (Han et al., 2022).

Our approach differs from prior work by using task-specific templates as verbal prompts for generation and classification which signal the model's objective. The model itself is self-controlling, generating novel data and retaining only the most convincing examples using a classification template to ensure semantic fidelity.

3 Method

In this section, we describe our novel self-controlled approach for text augmentation in text classification (STA). Figure 1 illustrates the workflow of STA and Algorithm 1 states STA in simple terms. At a high level, STA first finetunes a pre-trained sequence-to-sequence (seq2seq) model using a dataset which implicitly includes generation and classification tasks.

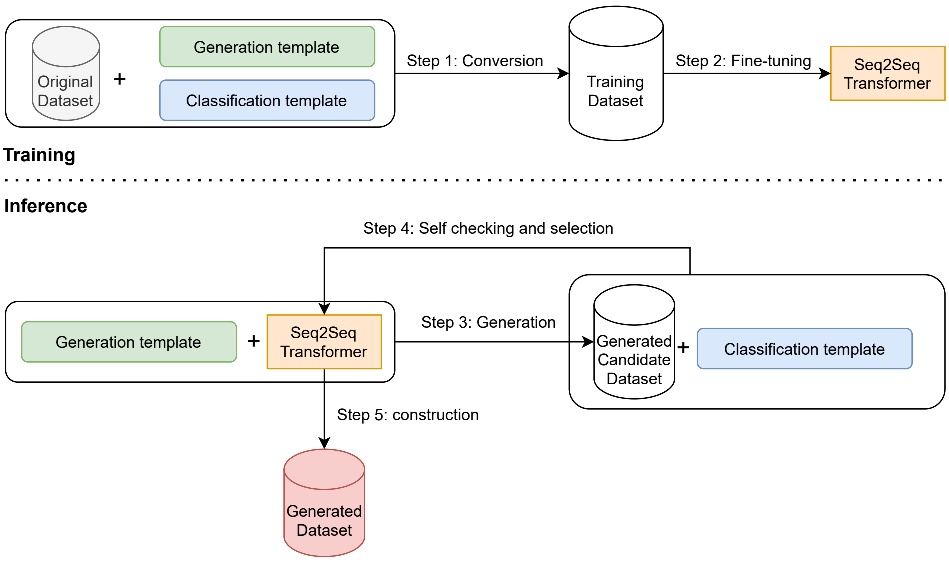


Figure 1: The architecture of our Self-controlled Text Augmentation approach (STA). The upper portion outlines the finetuning component of our method (**Training**), whilst the lower portion demonstrates our procedure for generating novel data (**Inference**). STA is highlighted by using the generation template and classification template for fine-tuning a seq2seq transformer model. The generation template is used for generating samples and the classification template is used for self-controlling and selecting the generated samples.

Algorithm 1 :Self-Controlled Text Augmentation

Require: Original dataset $\mathcal{D}_o$. Generative model M. Generation template $\mathcal{G}$. Classification template $\mathcal{C}$.

1: Convert $\mathcal{D}_o$ to training dataset $\mathcal{D}_t$ via $\mathcal{G}$ and $\mathcal{C}$.
2: Finetune M on $\mathcal{D}_t$ in a generation task and a classification task jointly to obtain M_t.
3: Use $\mathcal{G}$ and M_t to generate candidate dataset $\mathcal{D}_c$.
4: Apply M_t to do classification inference on $\mathcal{D}_c$ with $\mathcal{C}$ to select the most confident examples.
5: Form the final generated dataset $\mathcal{D}^*$ with the selected examples.

3.1 Pattern-Exploiting Training in seq2seq Models

PET is a finetuning technique for text classification tasks in masked language models, as demonstrated in (Schick and Schütze, 2021a). By converting inputs into cloze questions, PET enables accurate classification with minimal labeled data. We extend the principles of PET to seq2seq autoregressive models in this paper, presenting the theoretical process for prompting-based generation and our innovative self-controlled approach.

Consider a pretrained seq2seq autoregressive transformer model denoted as M (we use T5 (Raffel et al., 2020) in our experiments). This type of model comprises an encoder-decoder pair, where the encoder takes an input sequence s and generates a contextualized encoded sequence $\overline{s}$. The decoder then takes the encoded sequence and the current subsequence t: $\{t_1, t_2, ..t_{i-1}\}$ as input to compute the conditional distribution $p_M(t_i|t_{1:i-1}, \overline{s})$ for the subsequent token in the sequence. Given $\overline{s}$, the possible target sample (a sequence) t: $\{t_1, t_2, ..., t_m\}$ can be obtained via the factorization:

$$p_M(t_{1:m}|\overline{s}) = \prod_{i=1}^{m} p_M(t_i|t_{1:i-1}, \overline{s}) \qquad (1)$$

Let $\mathcal{D}_o = \{(x_i, y_i)\}|_{i=1}^{n}$ be a dataset for text classification where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{L}$ are text and label respectively. The goal is to produce a derived dataset $\mathcal{D}_t$ to finetune M and ensure it is primed for generating diverse and (label) faithful examples by leveraging a set of prompt templates.

Formally, a *template* is a function $T : V^* \times \mathcal{L} \rightarrow V^* \times V^*$ where V is the vocabulary of M and V^* denotes the set of finite sequences of symbols in V. Of course, the structure of these templates can be quite malleable. For example, a template could be constructed through intuitive human interpretable verbalizable terms, optimized automatically for the task, fine-tuned with soft prompts or made up of sequentially intuitive sub-prompts. Regardless of the approach, the process is the same.

Given a family of templates $\mathcal{T}$, we set $\mathcal{D}_t =$

$\mathcal{T}(\mathcal{D}_o) = \bigcup_{T \in \mathcal{T}} T(\mathcal{D}_o)$. That is, we convert each sample $(x_i, y_i) \in \mathcal{D}_o$ to $|\mathcal{T}|$ samples in the derived dataset D_t. In the field of synthetic data generation for low-resource scenarios, these templates generally belong to the collection of templates capable of generating novel examples. Crucially, we extend these templates to consider two types of template families: generation templates $\mathcal{G}$ and classification templates $\mathcal{C}$, such that $\mathcal{T} = \mathcal{C} \cup \mathcal{G}$. As we shall demonstrate, by carefully considering these templates, we can produce a dataset D_t (generated from these templates $\mathcal{T}$ applied to the dataset D) that is designed in such as way that the model can learn to directly optimize for key characteristics of the synthetic examples: High semantic fidelity and lexical diversity.

3.1.1 Generation templates

Though not exclusive to the field, these templates are commonplace within the synthetic data generation literature for creating novel training examples. Since our work focuses on encoder-decoder models, the templates take the form $g(x, y) = (f_s(x, y), f_t(x))$, where f_s and f_t denote functions that map a piece of text to a source sequence and target sequence respectively. Concretely, the source function f_s is a verbalizable mapping which depends on the text $x \in \mathcal{X}$ and label $y \in \mathcal{L}$, the latter of which conditions the model to align the generated text with the labels. The target function f_t on the other hand, represents the desired output of the model, which depends on the text, and typically corresponds to the identity function.

Diverse Generation. Without loss of generality, for a given downstream task {Task}, we could choose the primary template f_s = Description: $\{y_i\}$ $\{Task\}$. Text: as our source function and $f_t = \{x_i\}$ as the desired target for fine-tuning to facilitate the generation process, following previous work (Anaby-Tavor et al., 2020; Schick and Schütze, 2021b,a). Here the goal of $Task$ is to provide context about the dataset, since providing this sort of context helps when there are limited training examples (Schick and Schütze, 2021b). In this work, our goal is not only to generate novel sythetic examples for few-shot classification, but to generate a diverse range of these samples. To ensure the model produces lexically diverse text, we propose a novel generation strategy which additionally includes an auxiliary template for generation by including prior knowledge, partially inspired by work in state-of-the-art subprompt engineering (Han et al., 2022). Given some data point (x_i, y_i) we achieve diversity by modifying two components to our source and target functions.

- **Memory**: We add a previous example of text x_j which share the same label as an input to the source function, $j \in \mathbb{N}$ such that $y_j = y_i$.
- **Priming**: We instantiate the source function with some of the target output $x_i^{0-n}, n < |x_i| \in \mathcal{N}$, which further constrains the model to avoid the generation of non-factual hallucinations (Cao et al., 2022).

Concretely, we define a second auxiliary template function for generation $g'(x_i, x_j, y_i) = (f'_s(x_i, x_j, y_i), f'_t(x_i))$, with the source function $f'_s = f_s(x_j, y_i)$. Another text: $\{x_i^{0-2}\}$ and target function $f'_t = \{x_i^{3\cdots}\}$ where $y_j = y_i$. Intuitively, we use a previous example as prior knowledge before concatenating them with the new template to ensure the model produces distinct examples as opposed to repetitions. It's worth mentioning that the g' function can be employed multiple times to create various examples by sampling different texts during the conversion of a single training example (check Appendix B demonstrates how an original training sample is converted by the templates). For generation, we include both templates g and g' for tuning our model. These templates are further outlined in Table 1.

3.1.2 Classification templates

Classification has been employed as an additional processing step to filter synthetic examples which do not align with the generated label (Anaby-Tavor et al., 2020). In previous work, a separate network is trained using the original data to classify the examples, based on the intuition that checking the results is easier than producing new examples. One problem that emerges from adding a filter in low-resource settings is that it creates an additional layer of complexity within the system: Not only must the generator predict the correct label from limited data, but so must the classifier. These templates take the form $c(x, y) = (f_s(x), f_t(y))$ where f_t and f_s similarly denote the source sequence and target sequence functions respectively. In this case, the source functions are similar to the generation templates (the

Template		Source sequence (s)	Target sequence (t)	
Classification	Primary	c	Given {Task}: $\{\mathcal{L}\}$. Classify: $\{x_i\}$	$\{y_i\}$
	Auxiliary	c_{pos}	Text: $\{x_i\}$. Is this text about $\{y_i\}$ {Task}?	yes
	Auxiliary	c_{neg}	Text: $\{x_i\}$. Is this text about $\{\overline{y}_i\}$ {Task}?	no
Generation	Primary	g	Description: $\{y_i\}$ {Task}. Text:	$\{x_i\}$
	Auxiliary	g'	Description: $\{y_i\}$ {Task}. Text: $\{x_j\}$. Another text: $\{x_i^{0\text{-}2}\}$	$\{x_i^{3\cdots}\}$

Table 1: Prompt templates for training sequences conversion. "Task" refers to a simple keyword describing the dataset e.g. "Sentiment" or "Emotion" and $\mathcal{L}$ is the list of all class labels in the dataset. The symbol $\overline{y}_i$ in c_{neg} stands for any label in $\mathcal{L} \setminus \{y_i\}$, chosen randomly. In g', the x_j denotes another sample from the same class as x_i (i.e. $y_j = y_i$) chosen randomly.

text can be conditioned on the labels or independent), although the target function instead relates to the target label or some semantically compatible class. It is simple to translate the feed-forward approach into a primary template using verbalizations. In this case we set the source function as $f_s(x_i) =$ Given {Task}: $\{\mathcal{L}\}$. Classify: $\{x_i\}$ and target function as $f_t(y_i) = y_i$, with $\mathcal{L}$ providing context to the possible labels.

Semantic Fidelity. Although prompt-based tuning has proven to work better in limited data settings than simple feed-forward approaches (Le Scao and Rush, 2021), we further supplement the template dataset by generating multiple intuitive patterns following previous work (Schick and Schütze, 2021a). To achieve this, we supplement our base classification templates with two more auxiliary templates which we refer to as c_{pos} and c_{neg} in the vein of cloze-style questions. Concretely, we define $c_{pos} = (f_s(x_i), f_t(y_i))$ such that $f_s =$ Text: $\{x_i\}$. Is this text about $\{y_i\}$ {Task}? and $f_t=$ yes, with the goal of classifying whether the correct label conforms to the text. Furthermore, we generate a counter template $c_{neg} = (f_s(x_i), f_t(y_i))$ such that $f_s =$ Text: $\{x_i\}$. Is this text about $\{\hat{y_i}\}$ {Task}? and $f_t =$ no, $\hat{y_i} \sim \mathcal{L}\backslash\{y_i\}$, with the goal of determining that the incorrectly sampled label does not conform to the text. These templates are given in detail in Table 1.

Self-Checking. We note that these auxillary verbalizable patterns for classification are simply meant to supplement and do not represent the optimal solution for eliciting important knowledge from the network (Gao et al., 2021). We instead wish to avoid cascading errors between the generation and classification template: The classification network's performance should be within an acceptable tolerance. In order to extract synthetic examples with high levels of semantic alignment between the generated text and labels, we propose a novel strategy for controlled self-supervised data generation, which we refer to as *Self-Checking*. Different from previous work, we perform generation and classification filtering within a single unified neural framework. We hypothesise that this multiview learning process should allow the network to discover the semantic relationship between the labels and text, further preventing non-factual hallucinations of incorrect labels during the generation process.

3.2 Data Generation, Self-checking and Selection

We follow a two-step process: first we generate candidates and second we select a fraction of the candidates to be included as augmentations. This processes is conducted for each class separately so we may assume for the remainder of this section that we have fixed a label $y \in \mathcal{L}$.

That is, first, we generate $\alpha \times n_y$ samples where n_y is the original number of samples in $\mathcal{D}_o$ for label y and then select the top $\beta \times n_y$ samples ($\beta < \alpha$). In our experiments, we call β the *augmentation factor* and set $\alpha = 5 \times \beta$. Namely, our self-checking technique selects the top 20% of the candidate examples per class [1] to form the final generated D^* that is combined with the original dataset D_o for downstream model training.

For the generation task, we need to choose a prefix/source sequence s and proceed autoregressively using Equation 1. Referring back to Table 1, there are two choices g and g' that can be used to construct s. In this work, we employ g for generating

[1] This is based on our experimental search over {10%, 20%, 30%, 40%, 50%}.

examples because it allows for greater flexibility in generating diverse examples. We aim to generate as many diverse examples as possible at this stage (rather than selecting g', which requires a few initial words from an existing example as the context and can restrict the freedom of generating diverse examples). Nevertheless, all generated samples will be self-checked for semantic fidelity next.

Here we generate $\alpha \times n_y$ samples using the fine-tuned encoder-decoder model M_t where α is the times of the number of generated candidate examples to that of original examples.

We now possess a synthetic candidate dataset $\mathcal{D}_c^y = \{(x_i, y)\}|_{i=1}^{\alpha \times n_y}$ which we will refine using a self-checking strategy for selecting the generated samples based on the confidence estimated by the model M_t itself.

For each synthetic sample (x, y), we construct a source sequence using the classification template $c(x, y)$ as described in Table 1 to generate the source s. Given the source s, we define a score function u:

$$u(y|s) = \log p_{M_t}(\{y\}|\overline{s})$$

equivalently this is the *logit* computed by M_t for the sequence $\{y\}$. We then renormalize over the labels in $\mathcal{L}$ by applying a softmax over each of the scores $u(\cdot|s)$:

$$q(y|s) = \frac{e^{u(y|s)}}{\sum_{l \in \mathcal{L}} e^{u(l|s)}}$$

Finally, we rank the elements of $\mathcal{D}_c^y$ by the value of q and select the top $\beta \times n_y$ samples to form the dataset D_*^y and set $D_* = \bigcup_{y \in \mathcal{L}} D_*^y$

4 Experiments

Next, we conduct extensive experiments to test the effectiveness of our approach in low-data regimes. This section first describes the datasets choices, and then presents the baselines for comparison. Experimental details on how to train STA and evaluate it with the baselines in low-data settings can be found in the Appendix D.

4.1 Datasets

Following previous work in the augmentation literature (Kumar et al., 2020; Anaby-Tavor et al., 2020), two bench-marking datasets are used in our experiments: **SST-2** (Socher et al., 2013) and **TREC** (Li and Roth, 2002). We also include **EMOTION** (emotion classification) (Saravia et al., 2018) and **HumAID** (crisis tweets categorisation) (Alam et al., 2021) to extend the domains of testing STA's effectiveness. More information on the datasets can be found in Appendix C.

4.2 Baselines

We evaluate our novel strategy against a set of state-of-the-art techniques found within the literature. These approaches include a variety of augmentation procedures from rule-based heuristics to deep neural text generation. We compare STA to the augmentation techniques as they are directly related to our method in generating samples that can be used in our subsequent study for examining the quality of generated examples[2].

Baseline: No data augmentation is applied to the original training data.

EDA (Wei and Zou, 2019): Easy Data Augmentation involves applying local word-level changes to an existing example, such as synonym replacement and random insertion.

BT and BT-Hops (Edunov et al., 2018; Shleifer, 2019): Back-translation techniques involve translating from English to one (BT) or more randomly selected languages (BT-Hops) using a pre-trained translation model.

GPT-2 (Kumar et al., 2020) and GPT-2-λ (Anaby-Tavor et al., 2020): GPT-2[3] generates new examples conditioned on the label description and the first three words of an existing example. GPT-2-λ adds the LAMBDA technique, which selects generated examples based on the performance of the downstream classification model on the original training data.

CBERT (Wu et al., 2019): it is a strong word-replacement based method for text augmentation that replaces words in the original examples while conditioning on the labels.

BART-Span (Kumar et al., 2020):[4] it finetunes the large model BART (Lewis et al., 2020) based on the label names and the texts of 40% consecutive masked words to generate new examples.

5 Results and Discussion

5.1 Classification Tasks

Table 2 demonstrates the results of STA in comparison to baselines under low-data conditions for

[2]For a direct comparison between STA and existing non-augmentation few-shot baselines on downstream classification tasks, this refers to Appendix E.

[3]Licensing: Modified MIT License

[4]Licensing: Attribution-NonCommercial 4.0 International

Augmentation Method	5	10	20	50	100
Baseline (No Aug.)	56.5 (3.8)	63.1 (4.1)	68.7 (5.1)	81.9 (2.9)	85.8 (0.8)
EDA (Wei and Zou, 2019)	59.7 (4.1)	66.6 (4.7)	73.7 (5.6)	83.2 (1.5)	86.0 (1.4)
BT (Edunov et al., 2018)	59.6 (4.2)	67.9 (5.3)	73.7 (5.8)	82.9 (1.9)	86.0 (1.2)
BT-Hops (Shleifer, 2019)	59.1 (4.6)	67.1 (5.2)	73.4 (5.2)	82.4 (2.0)	85.8 (1.1)
CBERT (Wu et al., 2019)	59.8 (3.7)	66.3 (6.8)	72.9 (4.9)	82.5 (2.5)	85.6 (1.2)
GPT-2 (Kumar et al., 2020)	53.9 (2.8)	62.5 (3.8)	69.4 (4.6)	82.4 (1.7)	85.0 (1.7)
GPT-2-λ (Anaby-Tavor et al., 2020)	55.4 (4.8)	65.9 (4.3)	76.2 (5.6)	84.5 (1.4)	86.4 (0.6)
BART-Span (Kumar et al., 2020)	60.0 (3.7)	69.0 (4.7)	78.4 (5.0)	83.8 (2.0)	85.8 (1.0)
STA w/o Self-Checking	66.7 (5.0)	77.1 (4.7)	81.8 (2.1)	84.8 (1.0)	85.7 (1.0)
STA w/o Auxiliary Prompts	69.8 (4.9)	79.1 (3.4)	81.7 (4.5)	**86.0 (0.8)**	**87.5 (0.6)**
STA (ours)	**72.8 (6.2)**	**81.4 (2.6)**	**84.2 (1.8)**	**86.0 (0.8)**	87.2 (0.6)

Table 2: STA on **SST-2** in 5, 10, 20, 50, 100 examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

the **SST-2** classification task. The results of the remaining three classification tasks can be interpreted similarly; thus, they are presented in Appendix F, namely Table 7 for **EMOTION**, Table 8 for **TREC**, and Table 9 for **HumAID**, respectively. In all cases, our approach provides state-of-the-art performance for text augmentation across all low-resource settings. When a higher number of samples (50-100)[5] are used for training we see that STA is better, as in the cases of SST-2, EMOTION and HumAID tasks, or competitive, as in the case of TREC. Furthermore, we can see that STA is superior to other augmentation techniques when only a small number of examples are used to train the generator (5-10-20). In fact, STA on average demonstrates a difference of $+9.4\Delta$ and $+4.7\Delta$ when trained on only 5 and 10 samples per class respectively, demonstrating its ability to generate salient and effective training examples from limited amounts of data.

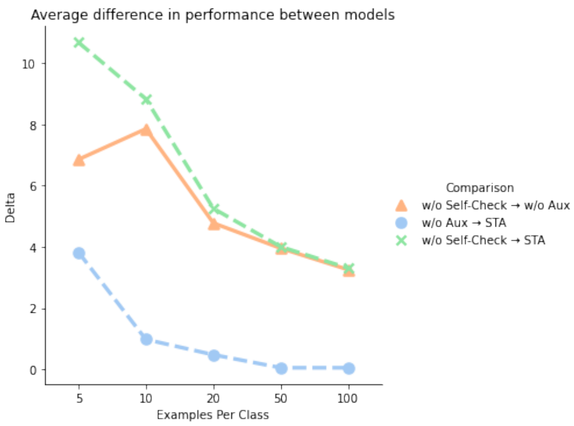
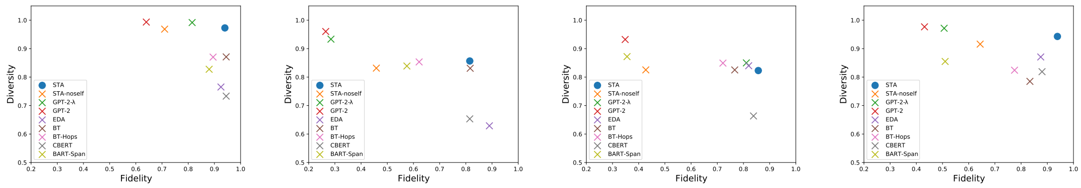


Figure 2: Graph showing the average difference between **STA w/o Self-Checking** to **STA w/o Auxiliary Prompts**, **STA w/o Auxiliary Prompts** to **STA** and **STA w/o Self-Checking** to **STA**, as the number of examples per class varies.

5.2 Ablation Studies: Self-Checking and Auxiliary Prompts

To demonstrate the importance of our self-checking procedure, we performed our empirical investigations on STA both with and without the self-checking step, denoted as **STA w/o Self-Checking** in Table 2, 7, 8 and 9. Furthermore, we investigate STA within a minimal template setting where we only include the templates c and g in Table 1, omitting our proposed auxiliary templates, denoted as **STA w/o Auxiliary Prompts**, to empirically separate the contribution of these components. Comparing our model with no self-checking (STA w/o Self-Checking) against other state-of-the-art approaches, we see that the model provides the best performance particularly when the data is more sparse (5-10-20), with the exclusion of TREC. However, when we add self-checking with only basic generation and classification templates (STA w/o Auxiliary Prompts), we see a significant improvement, indicating that self-checking more important to the downstream performance. We also compare the average difference between these models across all datasets with altering components in Figure 2. Looking at Figure 2 we see that the inclusion of self-checking provides the greatest increase in performance, while the contribution of our auxiliary prompts, including our novel generation template, decreases with larger examples per class. However, we note that the inclusion of both templates and self-checking provides the best performance, particularly in lower data regimes.

[5] We note that around 100 examples per class, all techniques tend to approximate no augmentation baselines, indicating that most likely constitute something more equivalent to full data training rather than a low-resource setting



(a) **SST-2** (b) **EMOTION** (c) **TREC** (d) **HumAID**

Figure 3: Diversity versus semantic fidelity of generated texts by various augmentation methods. The average scores over 10 runs are reported.

	SST-2	EMOTION	TREC	HumAID
Test	91.8	93.5	96.6	89.7

Table 3: Accuracy (in %) on test set predicted by BERT that is trained on the whole training data for measuring semantic fidelity.

5.3 Lexical Diversity and Semantic Fidelity

To further analyse the quality of the generated data, we measure its lexical diversity and semantic fidelity (i.e., its ability to align with the correct label). **Diversity** is assessed using the UNIQUE TRIGRAMS metric (Feng et al., 2020; Kumar et al., 2020), which calculates the ratio of unique tri-grams to total tri-grams in a population consisting of both original and generated training data. To coincide with the previous work (Kumar et al., 2020), **semantic fidelity** is determined by fine-tuning a “BERT-base-uncased” model on the 100% original training data for each classification task and measuring the accuracy of the generated data predictions by this model. Results are presented in Table 3. A higher score indicates better diversity or fidelity.

To present the quality of generated data in diversity and fidelity, we take the training data (10 examples per class) along with its augmented data ($\beta = 1$) for investigation. Figure 3 depicts the diversity versus semantic fidelity of generated data by various augmentation methods across three datasets. We find that generation-based approaches such as GPT-2 or GPT-2-λ, achieve strong diversity but less competitive fidelity. On the contrary, rule-based heuristics methods such as EDA perform well in retaining the semantic meaning but not in lexical diversity. The merit of STA is that it is good in both diversity and fidelity, as seen from its position at the top-right of Figure 3a, 3b, 3c and 3d. Finally, if we compare our STA approach with and without self-checking, we see that each approach produces highly diverse examples, although only self-checking STA retains a high level of semantic fidelity. Comparing with GPT-2 and GPT-2-λ — the other sample filtering approach — we see that the inclusion of a separate classifier results in an average increase of 18.3% in fidelity. However, if we compare our STA approach with and without self-checking, we see an average increase of 32.38% in fidelity, further demonstrating the validity of our join generation and classification approach as opposed to an independent classification module. As previously suggested, this ability to align the semantic content of generated examples with the correct label is the most probable reason for the increase in downstream classification performance when self-checking is employed. This supports the notion that our generation-based approach is able to produce novel data that is lexically diverse, whilst the self-checking procedure can ensure consistent label retention, which guarantees a high semantic fidelity in the generated examples[6].

6 Conclusion

We propose a novel strategy for text-based data augmentation that leverages prompt templates to generate training examples and ensure better label alignment. Our approach substantially outperforms the previous state-of-the-art on a variety of downstream classification tasks and across a range of low-resource scenarios. Furthermore, we provide an analysis of the lexical diversity and label consistency of generated examples, demonstrating that our approach produces uniquely varied training examples with more consistent label alignment than previous work. In the future, we hope to improve this approach in rich-data regime and extend it to other downstream natural language tasks.

[6]See also Appendix G for the demonstration of augmented examples.

References

Firoj Alam, Umair Qazi, Muhammad Imran, and Ferda Ofli. 2021. Humaid: Human-annotated disaster incidents data from twitter with deep learning benchmarks. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 15, pages 933–942.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Meng Cao, Yue Dong, and Jackie Chi Kit Cheung. 2022. Hallucinated but factual! inspecting the factuality of hallucinations in abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3340–3354.

Joe Davison, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. 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)*, pages 1173–1178.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *EMNLP*.

Steven Y Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. Genaug: Data augmentation for finetuning text generators. In *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 29–42.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. Ppt: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423.

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. Ptr: Prompt tuning with rules for text classification. *AI Open*, 3:182–192.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

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 *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pretrain, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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)*, pages 2463–2473.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402.

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 (Volume 1: Long Papers)*, pages 86–96.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Sam Shleifer. 2019. Low resource text classification with ulmfit and backtranslation. *arXiv preprint arXiv:1903.09244*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563.

Zirui Wang, Adams Wei Yu, Orhan Firat, and Yuan Cao. 2021. Towards zero-label language learning. *arXiv preprint arXiv:2109.09193*.

Jason 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)*, pages 6382–6388.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2022. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

A Limitations

Our work explores the possibility of data augmentation for boosting text classification performance when the downstream model is finetuned using pre-trained language models. The results show that STA consistently performs well across different bench-marking tasks using the same experimental setup, which addresses the limitation stated in the previous work (Kumar et al., 2020) calling for a unified data augmentation technique. However, similar to Kumar et al. (2020), although STA can achieve improved performance as the data size goes up to 100 examples per class in some cases (such as 100 examples per class in **EMOTION**, Table 7 and **HumAID**, Table 9), the absolute gain in performance plateaus when the training data becomes richer (such as 100 examples per class in **SST-2** and **TREC**). This suggests that it is challenging for STA to improve pre-trained classifier's model performance in more abundant data regimes.

Another important consideration is the choice of templates used in STA. Ablation experiments in Section 5.2 show that our chosen set of templates yields better performance than a 'minimal subset' consisting of the two simplest templates; the question as to how to choose optimal templates for this augmentation scheme remains unanswered. Hence, in future work, we will explore better methods for constructing the prompt templates, aiming to reduce the dependency on the manual work at this step.

B Template Example

Table 4 presents how an original training example is converted to multiple examples in STA using the prompt templates from Table 1.

C Datasets

Table 5 lists the basic information of the four datasets used in our experiments and they are shortly described as follows.

- **SST-2** (Socher et al., 2013) is a binary sentiment classification dataset that consists of movie reviews annotated with positive and negative labels.
- **EMOTION** (Saravia et al., 2018) is a dataset for emotion classification comprising short comments from social media annotated with six emotion types, such as, sadness, joy, etc.
- **TREC** (Li and Roth, 2002) is a dataset for question topic classification comprising questions across six categories including human, location, etc.
- **HumAID** (Alam et al., 2021) is a dataset for crisis messages categorisation comprising tweets collected during 19 real-world disaster events, annotated by humanitarian categories including rescue volunteering or donation effort, sympathy and support, etc.

D Training Details

When finetuning the generation model, we select the pre-trained T5 base checkpoint as the starting weights. For the downstream classification task, we finetune "bert-base-uncased"[7] on the original training data either with or without the augmented samples. Regarding the pre-trained models, we use the publicly-released version from the HuggingFace's transformers library (Wolf et al., 2019). For the augmentation factor (i.e., β in Section 3.2), the augmentation techniques including ours and the baselines are applied to augment 1 to 5 times of original training data. In the experiments, it is regarded as a hyper-parameter to be determined. Since our work focuses on text augmentation for classification in low-data settings, we sampled 5, 10, 20, 50 and 100 examples per class for each training dataset as per Anaby-Tavor et al. (2020). To alleviate randomness, we run all experiments 10 times so the average accuracy along with its standard deviation (std.) is reported on the full test set in the evaluation.

To select the downstream checkpoint and the augmentation factor, we select the run with the best performance on the development set for all methods. The hyper-parameters for finetuning the generation model and the downstream model are also setup based on the development set. Although using the full development set does not necessarily represent a real-life situation in low-data regime (Schick and Schütze, 2021a; Gao et al., 2021), we argue that it is valid in a research-oriented study. We choose to use the full development set since we aim to maximize the robustness of various methods' best performance given small training data available. As all augmentation methods are treated the same way, we argue this is valid to showcase the performance difference between our method and the baselines.

For all experiments presented in this work, we exclusively use *Pytorch*[8] for general code and *Huggingface*[9] for transformer implementations respectively, unless otherwise stated. In finetuning T5, we set the learning rate to 5×10^{-5} using Adam (Kingma and Ba, 2014) with linear scheduler (10% warmup steps), the training epochs to be 32 and batch size to be 16. At generation time, we use top-k ($k = 40$) and top-p ($p = 1.0$) sampling technique (Holtzman et al., 2019) for next token generation. In finetuning downstream BERT, the hyper-parameters are similar to those of T5 finetuning, although the training epoch is set to be 20. We set the training epochs to be as large as possible with the aim of finding the best model when trained on a small dataset, where the quality is based on performance on the development set. In our experiments, for a single run on all datasets, it takes around one day with a single Tesla P100 GPU (16GB) and thus estimated 10 days for 10 runs. To aid reproducibility, we will release our experimental code to the public at [10].

E Comparing to Few-shot Baselines

Since our work explores a text augmentation approach for improving text classification in low-data regime, it is also related to few-shot learning

[7] https://huggingface.co/bert-base-uncased

[8] https://pytorch.org/

[9] https://huggingface.co/

[10] https://github.com/wangcongcong123/STA

An example from **SST-2** a sentiment classification dataset where the classes ($\mathcal{L}$): negative, positive	
Text (x)	*top-notch action powers this romantic drama.*
Label (y)	*positive*
Converted examples by classification templates ($\mathcal{C}$: c, c_{pos} and c_{neg}): source(s), target(t)	
Given sentiment: negative, positive. Classify: top-notch action powers this romantic drama.	*positive*
Text: top-notch action powers this romantic drama. Is this text about positive sentiment?	*yes*
Text: top-notch action powers this romantic drama. Is this text about negative sentiment?	*no*
Converted examples by generation templates ($\mathcal{G}$: g and g'): source(s), target(t)	
Description: positive sentiment. Text:	*top-notch action powers this romantic drama.*
Description: positive sentiment. Text: top-notch action powers this romantic drama. Another text: spielberg 's realization of	*a near-future america is masterful .*
Description: positive sentiment. Text: top-notch action powers this romantic drama. Another text: a movie in	*which laughter and self-exploitation merge into jolly soft-porn 'em powerment . '*
Description: positive sentiment. Text: top-notch action powers this romantic drama . Another text: a tightly directed	*highly professional film that 's old-fashioned in all the best possible ways .*

Table 4: The demonstration of an example conversion by the prompt templates in Table 1 where the example's text is highlighted in blue and label is highlighted in red for readability.

Dataset	**# Train**	**# Dev**	**# Test**	**# Classes** (N)
SST-2	6,228	692	1,821	2
EMOTION	160,000	2,000	2,000	6
TREC	4,906	546	500	6
HumAID	40,623	5,913	11,508	8

Table 5: Datasets statistics

	SST-2	EMOTION	TREC
DART	66.5 (5.8)	26.7 (3.0)	74.0 (2.7)
LM-BFF	71.1 (9.5)	30.2 (3.8)	**77.1 (3.0)**
PET	56.7 (0.8)	28.4 (1.0)	69.1 (1.1)
STA (ours)	**81.4 (2.6)**	**57.8 (3.7)**	70.9 (6.6)

Table 6: The comparison between STA and few-shot baselines using 10 examples per class on **SST-2** and **EMOTION** and **TREC**. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

methods that use few examples for text classification. We further conduct an experiment to compare STA to three state-of-the-art few-shot learning approaches: PET (Schick and Schütze, 2021a), LM-BFF (Gao et al., 2021), and DART (Zhang et al., 2022). For fair comparison, we set the experiment under the 10 examples per class scenario with 10 random seeds ensuring the 10 examples per class are sampled the same across the methods. Besides, we use `bert-base-uncased`[11] as the starting weights of the downstream classifier. The results are shown in Table 6. We found that although STA loses the best score to DART and LM-BFF on the **TREC** dataset, it substantially outperforms the few-shot baselines on **SST-2** and **EMOTION**. This tells us that STA is a competitive approach for few-shot learning text classification.

F More Results of Classification Tasks

Table 7, Table 8 and Table 9 present the results of STA comparing to baselines in low-data settings for the **EMOTION**, **TREC** and **HumAID** classification tasks respectively.

G Demonstration

Table 10 and Table 11 demonstrate some original examples and augmented examples by different methods. In comparison, the examples generated by STA tend to be not only diverse but also highly label relevant (semantic fidelity).

[11] https://huggingface.co/bert-base-uncased

Augmentation Method	**5**	**10**	**20**	**50**	**100**
Baseline (No Aug.)	26.7 (8.5)	28.5 (6.3)	32.4 (3.9)	59.0 (2.6)	74.7 (1.7)
EDA	30.1 (6.2)	33.1 (4.3)	47.5 (5.0)	66.7 (2.7)	77.4 (1.8)
BT	32.0 (3.0)	37.4 (3.0)	48.5 (5.1)	65.5 (2.0)	75.6 (1.6)
BT-Hops	31.3 (2.6)	37.1 (4.6)	49.1 (3.5)	65.0 (2.3)	75.0 (1.5)
CBERT	29.2 (6.5)	32.6 (3.9)	44.1 (5.2)	62.1 (2.0)	75.5 (2.2)
GPT-2	28.4 (8.5)	31.3 (3.5)	39.0 (4.1)	57.1 (3.1)	69.9 (1.3)
GPT-2-λ	28.6 (5.1)	30.8 (3.1)	43.3 (7.5)	71.6 (1.5)	80.7 (0.4)
BART-Span	29.9 (4.5)	35.4 (5.7)	46.4 (3.9)	70.9 (1.5)	77.8 (1.0)
STA w/o Self-Checking	34.0 (4.0)	41.4 (5.5)	53.3 (2.2)	65.1 (2.3)	74.0 (1.1)
STA w/o Auxiliary Prompts	41.8 (6.1)	56.2 (3.0)	**64.9 (3.3)**	75.1 (1.5)	81.3 (0.7)
STA (ours)	**43.8 (6.9)**	**57.8 (3.7)**	64.1 (2.1)	**75.3 (1.8)**	**81.5 (1.1)**

Table 7: STA on **EMOTION** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

Augmentation Method	**5**	**10**	**20**	**50**	**100**
Baseline (No Aug.)	33.9 (10.4)	55.8 (6.2)	71.3 (6.3)	87.9 (3.1)	93.2 (0.7)
EDA	54.1 (7.7)	70.6 (5.7)	79.5 (3.4)	89.3 (1.9)	92.3 (1.1)
BT	56.0 (8.7)	67.0 (4.1)	79.4 (4.8)	89.0 (2.4)	92.7 (0.8)
BT-Hops	53.8 (8.2)	67.7 (5.1)	78.7 (5.6)	88.0 (2.3)	91.8 (0.9)
CBERT	52.2 (9.8)	67.0 (7.1)	78.0 (5.3)	89.1 (2.5)	92.6 (1.1)
GPT-2	47.6 (7.9)	67.7 (4.9)	76.9 (5.6)	87.8 (2.4)	91.6 (1.1)
GPT-2-λ	49.6 (11.0)	70.2 (5.8)	80.9 (4.4)	**89.6 (2.2)**	**93.5 (0.8)**
BART-Span	55.0 (9.9)	65.9 (6.7)	77.1 (5.5)	88.38 (3.4)	92.7 (1.6)
STA w/o Self-Checking	45.4 (3.2)	61.9 (10.2)	77.2 (5.5)	88.3 (1.2)	91.7 (0.8)
STA w/o Auxiliary Prompts	49.6 (9.0)	69.1 (8.0)	81.0 (5.9)	89.4 (3.0)	93.1 (0.9)
STA (ours)	**59.6 (7.4)**	**70.9 (6.6)**	**81.1 (3.9)**	89.1 (2.7)	93.2 (0.8)

Table 8: STA on **TREC** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

Augmentation Method	5	10	20	50	100
Baseline (No Aug.)	29.1 (6.6)	37.1 (6.4)	60.7 (4.0)	80.0 (0.9)	83.4 (1.0)
EDA	49.5 (4.5)	64.4 (3.6)	74.7 (1.5)	80.7 (1.0)	83.5 (0.6)
BT	45.8 (5.7)	59.1 (5.2)	73.5 (2.1)	80.4 (1.2)	83.1 (0.7)
BT-Hops	43.4 (6.4)	57.5 (5.2)	72.4 (2.8)	80.1 (1.1)	82.8 (1.4)
CBERT	44.8 (7.6)	59.5 (4.8)	73.4 (1.7)	80.3 (0.8)	82.7 (1.2)
GPT-2	46.0 (4.7)	55.7 (5.7)	67.3 (2.6)	77.8 (1.6)	81.1 (0.6)
GPT-2-λ	50.7 (8.6)	68.1 (6.2)	78.5 (1.3)	82.1(1.1)	84.2 (0.8)
BART-Span	42.4 (7.3)	58.6(7.0)	70.04 (3.7)	79.3 (1.4)	83.33 (0.9)
STA w/o Self-Checking	56.4 (7.0)	70.2 (4.3)	76.3 (3.3)	79.4 (4.5)	81.8 (1.3)
STA w/o Auxiliary Prompts	68.7 (10.9)	**77.6 (3.6)**	80.1 (1.7)	82.9 (1.6)	84.3 (0.7)
STA (ours)	**69.0 (3.9)**	75.8 (3.3)	**80.2 (1.6)**	**83.2 (0.5)**	**84.5 (1.1)**

Table 9: STA on **HumAID** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

Original training examples and augmented examples for "Sadness" of **EMOTION**	
Original	i sit here feeling blank about this i feel ashamed that i so readily turn it aside i feel positively ashamed when i look out of the window and see the state of things i had just lost my uncle i would be sad but i feel as if i am devastated i was feeling kind of discouraged because nothing happened
EDA	i sit here opinion blank about this i feel that ashamed i so readily turn it aside i feel positively ashamed when i look out of the window and construe the state of things i had just lost my uncle i would be pitiful but i feel as if i am devastated i happened feeling kind of discouraged because nothing was
GPT-2-λ	ive seen so many girls walk around feeling ashamed of their bodi ive got to admit that i feel a little weird for a moment seeing her standing in front of my face when i walk into the shop ive always wondered what im doing right now im feeling ive read many blogs about her and how much she hates those who don't admit to being kind or caring about others but instead blame them for not doing something about it ive never felt sympathetic towards people because of the way they look and act because of their skin to
STA-noself	i feel like the whole world is watching and feeling it's failing me i want people to know i am not alone i feel ashamed when i look out of the window and see the state of things i walked away feeling disappointed because i don t know the answer i drank some cold drink or find some ice dessert such as chendol or ice kacang
STA	i feel sad seeing people who have to work harder to cope i walked away feeling disappointed because i don t know the answer i was feeling sad seeing the state of things that i never did i really want to see if it lasted i feel sad seeing the state of things but the truth is im not sure how to express it gracefully i feel like the whole world is watching and feeling it's failing me

Table 10: The demonstration of original training examples and augmented examples for "sadness" of **EMOTION**. It is noted that the 5 augmented examples in each block are randomly selected instead of cherry-picked. This reveals some difference between the original training examples and the augmented examples by our STA and other methods (Here we use a rule-based heuristics method EDA, a generation-based method GPT-2-λ and STA-noself for comparison).

Original training examples and augmented examples for “missing or found people" of HumAID	
Original	UPDATE: Body found of man who disappeared amid Maryland flooding Open Missing People Search Database from Mati and Rafina areas #Greecefires #PrayForGreece #PrayForAthens @ThinBlueLine614 @GaetaSusan @DineshDSouza case in point, #California Liberalism has created the hell which has left 1000s missing 70 dead,... Heres the latest in the California wildfires #CampFire 1011 people are missing Death toll rises to 71 Trump blames fires on poor ... #Idai victims buried in mass grave in Sussundenga, at least 60 missing - #Mozambique #CycloneIdai #CicloneIdai
EDA	update flooding found of man who disappeared amid maryland boy open missing people search database from mati escape and rafina areas greecefires prayforgreece prayforathens created gaetasusan dineshdsouza hell in point california missing has thinblueline the case which has left s liberalism dead an countless people... heres blames latest in the california wildfires campfire people are missing death toll rises to trump more fires on poor... idai victims buried in mass grave in sussundenga at mozambique missing least cycloneidai cicloneidai
GPT-2-lambda	@KezorNews - Search remains in #Morocco after @deweathersamp; there has been no confirmed death in #Kerala #Cambodia - Search & Rescue is assisting Search & Rescue officials in locating the missing 27 year old woman who disappeared in ... @JHodgeEagle Rescue Injured After Missing Two Children In Fresno County #Florence #Florence Missing On-Rescue Teams Searching For Search and Rescue Members #Florence #Florence #DisasterInformer #E RT @LATTAODAYOUT: RT @HannahDorian: Search Continues After Disappearance of Missing People in Florida
STA-noself	Search Database from Matias, Malaysia, missing after #Maria, #Kerala, #Bangladesh #KeralaKerala, #KeralaFloods, ... RT @hubarak: Yes, I can guarantee you that our country is safe from flooding during the upcoming weekend! Previous story Time Out! 2 Comments The missing persons who disappeared amid Maryland flooding are still at large. More on this in the next article. the number of missing after #CycloneIdai has reached more than 1,000, reports CNN. RT @adriane@przkniewskiZeitecki 1 person missing, police confirm #CycloneIdai. #CicloneIdai
STA	The missing persons who disappeared amid Maryland flooding are still at large. More on this in the next article. Search Triangle County for missing and missing after #Maria floods #DisasterFire Just arrived at San Diego International Airport after #Atlantic Storm. More than 200 people were missing, including 13 helicopters ... Search Database contains information on missing and found people #HurricaneMaria, hashtag #Firefighter Were told all too often that Californians are missing in Mexico City, where a massive flood was devastating. ...

Table 11: The demonstration of original training examples and augmented examples for “missing or found people” of **HumAID**. It is noted that the 5 augmented examples in each block are randomly selected instead of cherry-picked. This reveals some difference between the original training examples and the augmented examples by our STA and other methods (Here we use a rule-based heuristics method EDA, a generation-based method GPT-2-λ and STA-noself for comparison).