# STA: Self-controlled Text Augmentation for Improving Text Classifications

Anonymous ACL submission

## 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 costly and time consuming and thus 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 in the corpus and thus can be well suited for faithful domain-specific generation. 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 and strong downstream performance . The trade off for diverse neurally-generated data is that semantic discrepancies can emerge which can cause samples to be misaligned with their appropriate label. 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. 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

Data augmentation for text classification has been widely developed in the literature. Zhang et al. (2015) demonstrated that replacing words or phrases with lexically similar words such as synonyms or hyponyms/hypernyms is an effective way to perform text augmentation with minimal loss of generality. The authors identify the target words according to a predefined geometric distribution and then replace words with their synonyms from a thesaurus. Similarly, Wei and Zou (2019) proposed EDA (*Easy Data Augmentation*) for text classification that generates new samples from the original training data with four simple operations; synonym replacement, random insertion, random swap, and random deletion, while Feng et al. (2020) further explores these substitution techniques, particularly for text generation. Wang and Yang (2015) instead exploit the distributional knowledge from word embedding models to randomly replace words or phrases with other semantically similar concepts. Kobayashi (2018) built upon this idea by replacing words based on the context of the sentence, which they achieve by sampling words from the probability distribution produced by a bi-directional LSTM-RNN language model at different word positions.

Back translation is another method that has shown to be effective for augmentation, particularly for transforming the structure of the text (Sennrich et al., 2016; Shleifer, 2019). Here, novel samples are generated by translating the original sentence to a predetermined language, before it is eventually translated back to the original target language. More recently, researchers have looked to capitalize on the success of pretrained transformer-based language models by performing conditional text augmentation to generate new sentences from the original training data. For example, (Wu et al., 2019) leveraged the masked language model of BERT conditioned on labeled prompts that are prepended to the text. Anaby-Tavor et al. (2020) was also successfully able to finetune GPT-2 with scarcely labeled training data to generate novel sentences of text, which improved performance on downstream classification tasks. Furthermore, the authors aimed to directly tackle the label misalignment problem by filtering noisy generated sentences using a jointly trained classifier, with some success. Similar work was performed by Kumar et al. (2020) who studied conditional text augmentation using a broader range of transformer-based pre-trained language models including autoregressive models (GPT-2), auto-encoder models (BERT), and seq2seq models (BART), the latter of which outperformed other data augmentation methods in a low-resource setting.

Recently, Wang et al. (2021) also 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. More recent and closely aligned with our work includes both LM-BFF (Gao et al., 2021) and DART (Zhang et al., 2022).

Unlike previous work, our novel approach can successfully generate diverse samples using task-specific templates — verbal prompts for generation and classification which signal the models objective. To ensure semantic fidelity, the model itself (self-controlling) is then used to both generate novel data and selectively retain only the most convincing examples using a classification template.

## 3 Method

In this section, we describe our 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 pretrained sequence-to-sequence (seq2seq) model using a dataset which implicitly includes generation and classification tasks. The generation task is then employed to generate new data, and the classification task is used for self-checking and selection for the final synthetic dataset.

**Algorithm 1** :Self-Controlled Text Augmentation

**Require:** Original dataset $\mathcal{D}_o$. Generation 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

Pattern-Exploiting Training, PET (Schick and Schütze, 2021a), is a finetuning technique for downstream text classification tasks in masked language models. The authors in (Schick and Schütze, 2021a) show PET allows accurate text classification with very few labeled examples by converting inputs into cloze questions. In this paper we adapt the principles of PET to seq2seq autoregressive models.

Let $M$ be a pretrained seq2seq autoregressive transformer model (for our experiments we have chosen T5 (Raffel et al., 2020)). Such models consist of an encoder-decoder pair; the encoder takes an input sequence $s$ and produces a contextualised encoding sequence $\overline{s}$. The encoded sequence and current subsequence $t\colon \{t_1, t_2, ..t_{i-1}\}$ are then used as the input for the decoder to compute the conditional distribution $p_M(t_i|t_{1:i-1}, \overline{s})$ for the next token in the sequence. It is the possible target sample (a sequence) $t\colon \{t_1, t_2, ..., t_m\}$ given $\overline{s}$ via the factorization:

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

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.

Formally, a *template* is a function $T : V^* \times \mathcal{L} \to V^* \times V^*$ where $V$ is the vocabulary of $M$ and $V^*$ denotes the set of finite sequences of symbols in $V$. 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 $D_t$. Table 1 lists all the templates we specifically designed for classification and generation purposes [1] and Table 7 (see Appendix A) demonstrates how this conversion is performed.

Crucially, we construct two types of template families: classification templates $\mathcal{C}$ and generation templates $\mathcal{G}$ and set $\mathcal{T} = \mathcal{C} \cup \mathcal{G}$.

**Classification templates** have the form $c(x, y) = (f_1(x, y), f_2(y))$ i.e. the text $x \in \mathcal{X}$ is not a part of the target output. **Generation templates** have the form $g(x, y) = (f_1(x, y), f_2(x))$ i.e. the label $y \in \mathcal{L}$ is not a part of the target output. Thus $D_t$ is designed so that our model can learn both how to generate a new piece of text of the domain conditioning on the label description as well as to classify a piece of text of the domain. With the dataset $D_t$ in hand, we proceed to finetune $M$ to obtain $M_t$, see 4.3 for details on training parameters. We next describe how to use $M_t$ for text generation and self-checking.

### 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 $y \in \mathcal{L}$.

The first objective is to generate $\alpha \times n_y$ samples where $n_y$ is the original number of samples in $\mathcal{D}_o$ for label $y$. To do so, all we need to is choose a prefix/source sequence $s$ and proceed autoregressively using Equation 1.

[1] We have a discussion on why we use this specific set of prompts in Section 7.

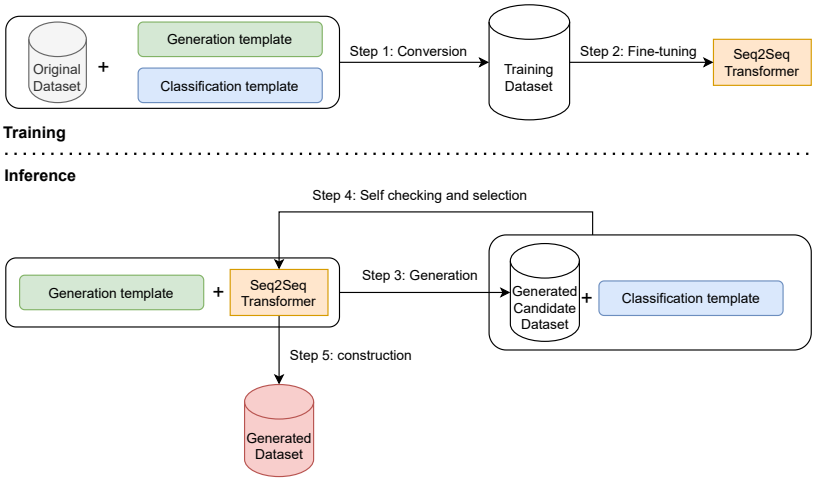



Figure 1: The architecture of our Self-controlled Text Augmentation approach (STA). The upper portion outlines the finetuning component of our method (**Training**), whilst the lower portion demonstrates our procedure for generating novel data (**Inference**). STA is highlighted by using the generation 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.

| **Template** | | **Source sequence** ($s$) | **Target sequence** ($t$) |
|---|---|---|---|
| Classification | $c_1$ | Given {Topic}: {$\mathcal{L}$}. Classify: {$x_i$} | {$y_i$} |
| | $c_2$ | Text: {$x_i$}. Is this text about {$y_i$} {Topic}? | yes |
| | $c_3$ | Text: {$x_i$}. Is this text about {$\overline{y}_i$} {Topic}? | no |
| Generation | $g_1$ | Description: {$y_i$} {Topic}. Text: | {$x_i$} |
| | $g_2$ | Description: {$y_i$} {Topic}. Text: {$x_j$}. Another text: {$x_i^{0\text{-}2}$} | {$x_i^{3\cdots}$} |

Table 1: Prompt templates. "Topic" 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 $c_3$ stands for any label in $\mathcal{L} \setminus \{y_i\}$, chosen randomly. In $g_2$, the $x_j$ denotes another sample from the same class as $x_i$ (i.e. $y_j = y_i$) chosen randomly.

Referring back to Table 1, we have two templates $g_1$ and $g_2$ to construct $s$. We choose $g_1$ over $g_2$ as the former only needs the label (we view the dataset description as a constant), i.e.

$$g_1(x, y) = (f_1(y), f_2(x)).$$

which gives the model greater freedom to generate diverse examples.

Thus we set $s = f_1(y)$ and generate $\alpha \times n_y$ samples using the finetuned model $M_t$ where $\alpha$ 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 template $c_1(x, y) = (h_1(x), \{y\})$, that is, we set $s = h_1(x)$. Given $s$ we define a score function $u$ in the same way as (Schick and Schütze, 2021a):

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

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 ($\beta < \alpha$) to form the dataset $D_*^y$ and set $D_* = \bigcup_{y \in \mathcal{L}} D_*^y$

In our experiments, we call $\beta$ the *augmentation factor* and set $\alpha = 5 \times \beta$. Namely, our self-checking technique selects the top 20% of the candidate examples per class [2] to form the final generated $D^*$ that is combined with the original dataset $D_o$ for downstream model training.

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

| 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-$\lambda$ (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-noself | 66.7 (5.0) | 77.1 (4.7) | 81.8 (2.1) | 84.8 (1.0) | 85.7 (1.0) |
| STA-twoprompts | 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.

## 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, and finally outlines model training and evaluation.

### 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 B.

### 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. We realise that our work is also related to few-shot learning approaches such as PET and LM-BFF that use few examples for text classification, we report the results of STA compared to them in Appendix C.

**Baseline (No Aug.)** uses the original training data as the downstream model training data. Namely, no augmentation is applied anywhere.

**EDA** (Wei and Zou, 2019) refers to easy data augmentation that transforms an existing example by applying local word-level changes such as synonym replacement, random insertion, etc.

**BT** and **BT-Hops** (Edunov et al., 2018; Shleifer, 2019) refers to back-translation techniques. The former is simply one step back translation from English to another language that is randomly sampled from the 12 Romance languages provided by the "opus-mt-en-ROMANCE" model[3] from the transformers library (Wolf et al., 2019). The latter adds random 1 to 3 extra languages in the back translation using the same model.

**GPT-2** [4] is a deep learning method using GPT-2 for augmentation. Following (Kumar et al., 2020), we finetune a GPT-2 base model on the original training data and then use it to generate new examples conditioning on both the label description and the first three words of an existing example.

**GPT-2-$\lambda$** is similar to GPT-2 with the addition of the LAMBDA technique from Anaby-Tavor et al. (2020). LAMBDA first finetunes the downstream classification model on the original training data and then use it to select the generated examples by GPT-2.

**CBERT** (Wu et al., 2019) is a strong word-replacement based method for text augmentation. It relies on the masked language model of BERT to obtain new examples by replacing words of the original examples conditioning on the labels.

**BART-Span** (Kumar et al., 2020) [5] uses the Seq2Seq BART model for text augmentation. Previously, it was found to be a competitive technique

[3] https://huggingface.co/Helsinki-NLP/opus-mt-en-ROMANCE

[4] Licensing: Modified MIT License

[5] Licensing: Attribution-NonCommercial 4.0 International

for data augmentation using BERT for classification (the sort of large-scale language models finetuning for classification) in low-data regimes. It is implemented as described in Kumar et al. (2020) that finetunes the BART large model conditioning on the label names and the texts of 40% consecutive masked words.

### 4.3 Training and Evaluation

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"[6] 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., $\beta$ 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. More information on training and evaluation refers to Appendix D.

## 5 Results and Discussion

### 5.1 Classification Tasks

The results on **SST-2** (Table 2), **EMOTION** (Table 3), **TREC** (Table 4) and **HumAID** (Table 5) classification tasks all demonstrate the effectiveness of our augmentation strategy. 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) 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.

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

### 5.2 Ablation Studies: Self-checking and 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. The results without self-checking are shown at the bottom of the tables for **SST-2** (Table 2), **EMOTION** (Table 3), **TREC** (Table 4) and **HumAID** (Table 5), denoted as "STA-noself". We see that our approach demonstrates considerable improvements when the self-checking step is added across all tasks and training sample sizes, further supporting our augmentation technique. In fact, the difference between the two settings is considerable, with an average increase of $+9.3\Delta$ across all tasks and training samples sizes. We hypothesize that the self-checking step more reliably controls the labels of the generated text, which greatly improves training stimulus and thus the performance on downstream tasks.

Of course, there are many possible choices for templates and permutations of template procedures. To further support the use of our multiple prompt templates used in STA (see Table 1), we conduct another ablation run for this purpose, denoted as "STA-twoprompts" at the bottom of the tables. These templates, one for classification ($c_1$) and one for generation ($g_1$), represent a minimalistic approach for performing generation-based augmentation with self-checking without the additional templates outlined in Table 1. The results show that the multiple templates used for STA provide additional improvements to the downstream tasks, especially in low-data settings.

To further analyse the quality of the generated data, we measure the diversity of the data, indicated by its lexical variation, and its ability to align the text with the correct label (semantic fidelity). The measurements for each are described as follows.

### 5.3 Lexical Variation and Semantic Fidelity

**Generated Data Diversity.** The metric we used for evaluating diversity is UNIQUE TRIGRAMS (Feng et al., 2020; Kumar et al., 2020). It is determined by calculating the unique tri-grams divided by the total tri-grams in a population. As we aim to examine the difference between the generated data and the original data, the population consists of both the original and generated training

| **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-$\lambda$ | 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-noself | 34.0 (4.0) | 41.4 (5.5) | 53.3 (2.2) | 65.1 (2.3) | 74.0 (1.1) |
| STA-twoprompts | 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)** |

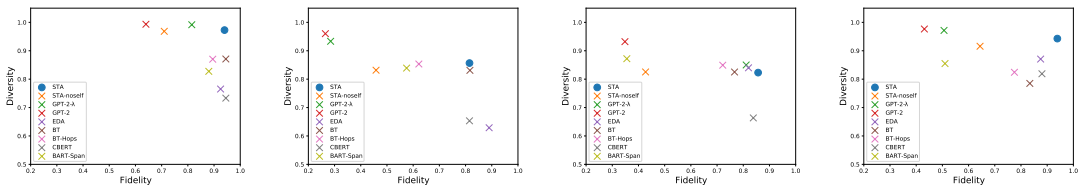
Table 3: STA on **EMOTION** in $5, 10, 20, 50, 100$ examples per class. The results are reported as average (std.) accuracy (in %) based on 10 random experimental runs. Numbers in **bold** indicate the highest in columns.

| **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-$\lambda$ | 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-noself | 45.4 (3.2) | 61.9 (10.2) | 77.2 (5.5) | 88.3 (1.2) | 91.7 (0.8) |
| STA-twoprompts | 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 4: 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-$\lambda$ | 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-noself | 56.4 (7.0) | 70.2 (4.3) | 76.3 (3.3) | 79.4 (4.5) | 81.8 (1.3) |
| STA-twoprompts | 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 5: 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.




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

Figure 2: 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 6: Accuracy (in %) on test set predicted by BERT that is trained on the whole training data for measuring semantic fidelity.

data. For this metric, a higher score indicates better diversity.

**Generated Data Fidelity.** The semantic fidelity is measured by evaluating how well the generated data retains the semantic meaning of its label. As per Kumar et al. (2020), we measure it by first finetuning a "BERT-base-uncased " on the 100% of original training data of each classification task. The performance of the classifier on the test set is reported in Table 6. After the finetuning, to measure the generated data fidelity, we use the finetuned classifier to predict the labels for the generated data and use the accuracy between its predicted labels and its associated labels as the metric for fidelity. Hence, a higher score indicates better 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 2 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-$\lambda$, 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 2a, 2b, 2c and 2d. 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. 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[7].

## 6 Conclusion

We propose a novel strategy for text-based data augmentation that uses pattern-exploiting training 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.

## 7 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 3 and **HumAID**, Table 5), 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.

[7] See also Appendix E 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.

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)*.

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.

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. Pre-train, 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.

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.

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.

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 Template Example

Table 7 presents how an original training example is converted to multiple examples in STA using the prompt templates from Table 1.

## B Datasets

Table 8 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.

## C 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 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`[8] as the starting weights of the downstream classifier. The results are shown in Table 9. 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.

## D Training Details

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

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

| 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: 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: 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 7: 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 8: 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 9: 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.

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*[9] for general code and *Huggingface*[10] for transformer implementations respectively, unless otherwise stated. In finetuning T5, we set the learning rate to $5 \times 10^{-5}$ 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 [11].

[9] https://pytorch.org/
[10] https://huggingface.co/
[11] Removedforanonymousreview

## E 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).

| Original training examples and augmented examples for “Sadness" of **EMOTION** | |
|---|---|
| Original | i sit here feeling blank about this<br>i feel ashamed that i so readily turn it aside<br>i feel positively ashamed when i look out of the window and see the state of things<br>i had just lost my uncle i would be sad but i feel as if i am devastated<br>i was feeling kind of discouraged because nothing happened |
| EDA | i sit here opinion blank about this<br>i feel that ashamed i so readily turn it aside<br>i feel positively ashamed when i look out of the window and construe the state of things<br>i had just lost my uncle i would be pitiful but i feel as if i am devastated<br>i happened feeling kind of discouraged because nothing was |
| GPT-2-$\lambda$ | ive seen so many girls walk around feeling ashamed of their bodi<br>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<br>ive always wondered what im doing right now im feeling<br>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<br>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<br>i want people to know i am not alone<br>i feel ashamed when i look out of the window and see the state of things<br>i walked away feeling disappointed because i don t know the answer<br>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<br>i walked away feeling disappointed because i don t know the answer<br>i was feeling sad seeing the state of things that i never did i really want to see if it lasted<br>i feel sad seeing the state of things but the truth is im not sure how to express it gracefully<br>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-$\lambda$ 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<br>Open Missing People Search Database from Mati and Rafina areas #Greecefires #PrayForGreece #PrayForAthens<br>@ThinBlueLine614 @GaetaSusan @DineshDSouza case in point, #California Liberalism has created the hell which has left 1000s missing 70 dead,...<br>Heres the latest in the California wildfires #CampFire 1011 people are missing Death toll rises to 71 Trump blames fires on poor ...<br>#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<br>open missing people search database from mati escape and rafina areas greecefires prayforgreece prayforathens<br>created gaetasusan dineshdsouza hell in point california missing has thinblueline the case which has left s liberalism dead an countless people...<br>heres blames latest in the california wildfires campfire people are missing death toll rises to trump more fires on poor...<br>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<br>#Cambodia - Search & Rescue is assisting Search & Rescue officials in locating the missing 27 year old woman who disappeared in ...<br>@JHodgeEagle Rescue Injured After Missing Two Children In Fresno County<br>#Florence #Florence Missing On-Rescue Teams Searching For Search and Rescue Members #Florence #Florence #DisasterInformer #E<br>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, ...<br>RT @hubarak: Yes, I can guarantee you that our country is safe from flooding during the upcoming weekend!<br>Previous story Time Out! 2 Comments<br>The missing persons who disappeared amid Maryland flooding are still at large. More on this in the next article.<br>the number of missing after #CycloneIdai has reached more than 1,000, reports CNN.<br>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.<br>Search Triangle County for missing and missing after #Maria floods #DisasterFire<br>Just arrived at San Diego International Airport after #Atlantic Storm. More than 200 people were missing, including 13 helicopters ...<br>Search Database contains information on missing and found people #HurricaneMaria, hashtag #Firefighter<br>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-$\lambda$ and STA-noself for comparison).