Text AutoAugment:
Learning Compositional Augmentation Policy for Text Classification

Shuhuai Ren[1], Jinchao Zhang[3], Lei Li[1], Xu Sun[1,2], Jie Zhou[3]

[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]Center for Data Science, Peking University
[3]Pattern Recognition Center, WeChat AI, Tencent Inc, China
`{shuhuai_ren, lilei}@stu.pku.edu.cn, xusun@pku.edu.cn`
`{dayerzhang, jiezhou}@tencent.com`

Abstract

Data augmentation aims to enrich training samples for alleviating the overfitting issue in low-resource or class-imbalanced situations. Traditional methods first devise task-specific operations such as Synonym Substitute, then preset the corresponding parameters such as the substitution rate artificially, which require a lot of prior knowledge and are prone to fall into the sub-optimum. Besides, the number of editing operations is limited in the previous methods, which decreases the diversity of the augmented data and thus restricts the performance gain. To overcome the above limitations, we propose a framework named **T**ext **A**uto**A**ugment (TAA) to establish a *compositional* and *learnable* paradigm for data augmentation. We regard a combination of various operations as an augmentation policy and utilize an efficient Bayesian Optimization algorithm to automatically search for the best policy, which substantially improves the generalization capability of models. Experiments on six benchmark datasets show that TAA boosts classification accuracy in low-resource and class-imbalanced regimes by an average of 8.8% and 9.7%, respectively, outperforming strong baselines.[1]

1 Introduction

Model performance on Natural Language Processing (NLP) tasks, such as text classification, often heavily depends on the size and the quality of the training data. However, it is time-consuming and labor-intensive to obtain sufficient training instances, and models face low-resource regimes most of the time. Data augmentation (Simard et al., 1996; Szegedy et al., 2015; Wei and Zou, 2019) aims to enlarge the training dataset by synthesizing additional distinct and label-invariant instances based on raw instances to improve the model per-

---

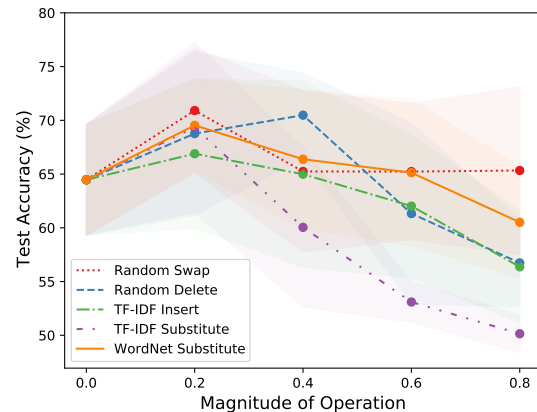[1]Our code is available at `https://github.com/lancopku/text-autoaugment`

Figure 1: Test accuracy on IMDB dataset with different editing operations and parameters for data augmentation. The operations and parameters without elaborate selection lead to a degrade of performance.

formance. Data augmentation approaches demonstrate their superiority in many scenarios, especially when data resources are insufficient (Hu et al., 2019b; Anaby-Tavor et al., 2020; Chen et al., 2021) and class-imbalanced (Ren et al., 2018; Hu et al., 2019b).

Previous data augmentation methods can be roughly divided into two categories: generation-based (Sennrich et al., 2016; Imamura et al., 2018; Kobayashi, 2018; Anaby-Tavor et al., 2020; Quteineh et al., 2020) and editing-based methods (Wei and Zou, 2019; Xie et al., 2020). Generation-based methods utilize conditional generation models (Sutskever et al., 2014; Ott et al., 2019) to synthesize the paraphrases (Kumar et al., 2019; Hu et al., 2019a) of the original sentences, which have advantages in instance fluency and label preservation but suffer from the heavy cost of model pre-training and decoding. Editing-based methods instead apply label-invariant sentence editing operations (swap, delete, etc.) on the raw instance, which are simpler and more efficient in practice. However, the editing-based methods are
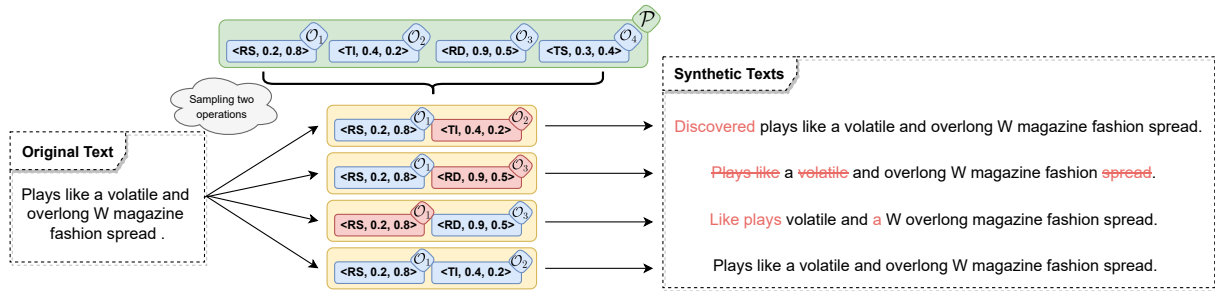
Figure 2: The structure and usage of augmentation policy $\mathcal{P}$. A policy $\mathcal{P}$ consists of N atomic editing operations $\mathcal{O} = \langle t, p, \lambda \rangle$. Given a text, we randomly pick N^* operations from $\mathcal{P}$ then apply the operations sequentially. In this case, $N = 4$ and $N^* = 2$. Red operation $\mathcal{O}$ indicates it is finally applied to the text according to probability p. The modification to the original text cased by it is also showed in Red. We use **RS** (Random Swap), **RD** (Random Delete), **TI** (TF-IDF Insert), **TS** (TF-IDF Substitute) for abbreviation. Best viewed in color.

sensitive to the preset hyper-parameters including the type of the applied operations and the proportion of words to be edited. As shown in Figure 1, we probe the classification accuracy on IMDB dataset (Maas et al., 2011) with different editing operations and magnitudes (proportion of edited words). We find that inappropriate hyper-parameter settings, e.g, TF-IDF Substitute (Xie et al., 2020) with a magnitude larger than 0.4, lead to inferior results. Therefore, heuristic hyper-parameter setting is prone to fall into the sub-optimum and lacks effectiveness. Besides, most of the editing-based methods (Kobayashi, 2018; Hu et al., 2019b; Sennrich et al., 2016; Imamura et al., 2018; Andreas, 2020) only apply a single operation on the sentence once a time, which restrict the diversity of the augmented dataset and thus limit the performance gain.

To overcome the limitations above, we propose a framework named **T**ext **A**uto**A**ugment (TAA) to establish a *learnable* and *compositional* paradigm for the data augmentation. Our goal is to automatically learn the optimal editing-based data augmentation policy for obtaining a higher quality augmented dataset and thus enhancing the target text classification model. We design an augmentation policy as a set of various editing operations: policy $= \{\mathrm{op}_1, \cdots, \mathrm{op}_N\}$ and each operation is defined with a parameter triplet: $\mathrm{op} = \langle \mathrm{type}\, t, \; \mathrm{probability}\, p, \; \mathrm{magnitude}\, \lambda \rangle$. Such a compositional structure allows more than one operation to be applied to the original sentence and can improve the the diversity of the synthetic instances. In conclusion, a policy solution consists of two kinds of knowledge to learn: the operation set and the editing parameters for each operation. To search the optimal policy, we propose a novel objective function and utilize the sequential Model-based

Global Optimization (SMBO) (Bergstra et al., 2011), an efficient and widely used method in AutoML (Yao et al., 2018), to learn the optimal configuration of the compositional policy.

Given a target dataset, our algorithm learns an augmentation policy automatically and adaptively. The implementation based on distributed learning frameworks can efficiently obtain promising results in several GPU hours. To summarize, our contribution is two-fold:

- We present a learnable and compositional framework for data augmentation. Our proposed algorithm automatically searches for the optimal compositional policy, which improves the diversity and quality of augmented samples.

- In low-resource and class-imbalanced regimes of six benchmark datasets, TAA significantly improves the generalization ability of deep neural networks like BERT and effectively boosts text classification performance.

2 Text AutoAugment

In this section, we introduce the Text AutoAugment framework to search for the optimal augmentation policy automatically. Our augmentation policy is composed of various operations and forms a hierarchical structure, which is first detailed in Section 2.1. Then we give an overview of the proposed learnable augmentation method and establish a global objective function for it (Section 2.2). The specific policy optimization algorithm is presented at last (Section 2.3).

2.1 Compositional Augmentation Policy

To generate a much broader set of augmentations, we introduce a compositional policy instead of a single operation. Figure 2 shows the structure and usage of the policy $\mathcal{P}$. Specifically, our policy is a set of N various editing operations:

$$\mathcal{P} = \{\mathcal{O}_1, \cdots, \mathcal{O}_i, \cdots, \mathcal{O}_N\}$$

The operation $\mathcal{O}$ is an atomic component and is responsible for applying an editing transformation on a text x to synthesize an augmented instance x_{aug}. Each operation $\mathcal{O}_i$ is defined as a triplet:

$$\mathcal{O}_i = \langle t_i, p_i, \lambda_i \rangle$$

with three parameters: (1) Type $t \in \{$*Random Swap, Random Delete, TF-IDF Insert, TF-IDF Substitute, WordNet Substitute*$\}$.[2] (2) Probability $p \in [0, 1]$ of being applied. (3) Magnitude $\lambda \in [0, 0.5]$ which determines the proportion of the words to be edited.

The augmented text after an operation can be formulated as:

$$\mathcal{O}_i\left(x;\ \langle t_i, p_i, \lambda_i \rangle\right) = \begin{cases} \mathcal{O}_i\left(x; \langle t_i, \lambda_i \rangle\right) & \text{with } p_i \\ x & \text{with } 1 - p_i \end{cases} \quad (1)$$

It means that for a given text x, it has probability p_i to be applied with the operation $\mathcal{O}_i$ for synthesizing the transformed text $x_{\text{aug}} = \mathcal{O}_i\left(x; \langle t_i, \lambda_i \rangle\right)$ under the magnitude λ_i, while has probability $1 - p_i$ to remain identical, i.e., $x_{\text{aug}} = x$.

To further increase the diversity of augmented data and enlarge the support of the training distribution, our policy applies more than one operation to the original sentence in a recursive way. Specifically, we randomly sample N^* editing operations from the policy $\mathcal{P}$ and apply them to a given text consecutively. The N^* atomic operations can be combined as a compositional operation with the recursive depth equals N^*. For example, when $N^* = 2$ and the sampled operations are $[\mathcal{O}_i, \mathcal{O}_j]$, the final augmented instance can be denoted as:

$$x_{\text{aug}} = \mathcal{O}_j\left(\mathcal{O}_i\left(x; \langle t_i, p_i, \lambda_i \rangle\right); \langle t_j, p_j, \lambda_j \rangle\right).$$

In other words, one policy can synthesize up to $A_N^{N^*} \times 2^{N^*}$ augmented instances, where A denotes a permutation. Note that our policy is determined by only $N \times 3$ parameters, which does NOT cause

---

[2]Please refer to Appendix A for the detailed description of each type.

the problem of search space explosion. As discussed before, the setting of these parameters in a policy has a great impact on the quality of augmented data and the model performance, which motivates us to devise a learnable framework for automatically selecting the optimal parameters instead of a naive grid search or cumbersome manual tuning.

2.2 Learnable Data Augmentation Policy

In this subsection, we first review the data augmentation and model training with the traditional objective function. We then propose a new objective function to learn the optimal policy for obtaining a augmented dataset with higher quality and improving the model performance.

Given an input space $\mathcal{X}$ and output space $\mathcal{Y}$ of the text classification task, a model f is responsible for learning a mapping from input texts $x \in \mathcal{X}$ to target labels $y \in \mathcal{Y}$. In some scenarios, the training set $\mathcal{D}_{\text{train}}$ is extremely small or imbalanced, which leads to a large generalization error on the test set. Therefore, data augmentation is incorporated as an implicit regularizer (Hernández-García and König, 2018) to help models learn better patterns (Simard et al., 1996) and further improve the generalization ability. Let $\mathcal{D}_{\text{aug}}(\mathcal{P})$ be the augmented dataset containing both training set and the synthetic data generated by the policy $\mathcal{P}$, the loss function $\mathcal{L}$ of model training on the augmented dataset can be formulated as a sum of instance-level loss l such as cross-entropy:

$$\mathcal{L} = \sum_{(x,y) \in \mathcal{D}_{\text{aug}}(\mathcal{P})} l\left(f\left(x\right), y\right). \quad (2)$$

In traditional methods, the parameters of data augmentation are preset before training then tuned on the validation set $\mathcal{D}_{\text{val}}$, which is similar to the paradigm of hyper-parameters tuning. For this reason, we cast the problem of augmentation policy optimization as the *Combined Algorithm Selection and Hyper-parameter (CASH)* optimization problem (Thornton et al., 2013; Feurer et al., 2019) in AutoML (Yao et al., 2018). Formally, let $\mathbb{F}$ and $\mathbb{P}$ be the search space of models and policies, respectively. Each model f is trained on $\mathcal{D}_{\text{aug}}(\mathcal{P})$ augmented by the policy $\mathcal{P}$. We propose a novel metric to measure the loss of a policy and a model:

$$\mathcal{J} := \mathcal{J}\left(f, \mathcal{D}_{\text{aug}}(\mathcal{P}), \mathcal{D}_{\text{val}}\right) \quad (3)$$

Here, $\mathcal{J}\left(f, \mathcal{D}_{\text{aug}}(\mathcal{P}), \mathcal{D}_{\text{val}}\right)$ denotes the loss that the model f achieves on the validation set $\mathcal{D}_{\text{val}}$
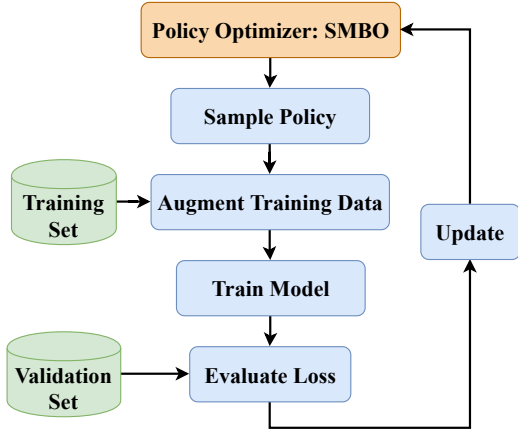
Figure 3: The overview of the optimization procedure in the Text AutoAugment algorithm. In each iteration, the optimizer samples a policy and trains a corresponding model on the augmented training set. After that, the loss on the validation set is calculated to update the policy optimizer then execute the next iteration.

after trained on the augmented set $\mathcal{D}_{\text{aug}}(\mathcal{P})$. It quantifies the generalization ability of the model after augmentation thus reflects the quality of the augmented data. Consequently, the objective function of our policy optimization can be written as

$$f^*, \mathcal{P}^* = \arg\min_{\substack{f \in \mathbb{F} \\ \mathcal{P} \in \mathbb{P}}} \mathcal{J}(f, \mathcal{D}_{\text{aug}}(\mathcal{P}), \mathcal{D}_{\text{val}}). \quad (4)$$

Through minimizing the validation loss of the model, we can find the optimal configuration of the policy $\mathcal{P}$ and improve the quality of the augmented data.

2.3 Augmentation Policy Optimization

Like the optimization of hyper-parameters of a deep learning model, our objective function for policy optimization depends on the validation loss, which can not be solved by gradient-based methods such as back-propagation. To tackle this problem, we introduce *Sequential Model-based Global Optimization* (SMBO) (Bergstra et al., 2011), a kind of Bayesian optimizer widely used in AutoML, to our TAA framework. Our optimization procedure is illustrated in Figure 3. In a nutshell, the SMBO optimizer builds a probability model of the objective function as a surrogate and uses it to samples the most promising policy, then evaluates the policy in the true objective function. In practice, we use Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) as a surrogate agent M to model the function between the policy and the objective loss in Eq. 7.

Algorithm 1: TAA Algorithm

Input: Number of iterations T
Input: Training set D_{train}
Input: Validation set D_{val}
$\mathcal{H} \leftarrow \emptyset$ // Initialize history
for $i = 1$ *to* T **do**
  $\mathcal{P}_i \leftarrow \arg\max_{\mathcal{P}} \text{EI}(\mathcal{P})$ // Eq. 8
  Train f_i // Eq. 2
  Evaluate $\mathcal{J}_i$ // Eq. 7
  $\mathcal{H} \leftarrow \mathcal{H} \cup (f_i, \mathcal{P}_i, \mathcal{J}_i)$
  Update surrogate model M to fit $\mathcal{H}$
end
$f^*, \mathcal{P}^* = \arg\min_{\substack{f \in \mathcal{H} \\ \mathcal{P} \in \mathcal{H}}} \mathcal{J}$ // Eq. 4
return f^* *and* $\mathcal{P}^*$

This process is carried out in an iterative manner. At each iteration, the SMBO optimizer samples an augmentation policy $\mathcal{P}_i$ to synthesize the augmented set $\mathcal{D}_{\text{aug}}(\mathcal{P}_i)$, and then trains a model f_i based on it. The loss $\mathcal{J}_i$ of the policy $\mathcal{P}_i$ is then evaluated by Eq. 7 and merged into the observation history $\mathcal{H}$ to help update the surrogate model.

The specific updating procedure of TPE is introduced in Appendix B. After that, we employ the following Expected Improvement (EI) criterion as an acquisition function to sample the next promising policy:

$$\text{EI}(\mathcal{P}) = \mathbb{E}\left[\max\left(\mathcal{J}^\dagger - \mathcal{J}, 0\right)\right] \quad (5)$$

Here, $\mathcal{J}^\dagger$ is a threshold calculated by the observation history and the surrogate model. Eq. 8 is the expectation under the surrogate model that the loss $\mathcal{J}$ of a policy will exceed (negatively) the threshold $\mathcal{J}^\dagger$. Note that our target is to find the policy that minimizes the loss $\mathcal{J}$, the policy that maximizes the expected improvement will be chosen in the next iteration. The TAA framework is summarized in Algorithm 1.[3]

3 Experiments

3.1 Benchmark Datasets

We conduct experiments on six benchmark datasets, including IMDB (Maas et al., 2011), SST-2, SST-5 (Socher et al., 2013), TREC (Li and Roth, 2002), YELP-2 and YELP-5 (Zhang et al., 2015). The statistics of the datasets are listed in Appendix C.

3.2 Baselines

We compare our TAA method with the following representative baselines:

---

[3]Due to limited space, please refer to Appendix B for the detailed description of the surrogate model update and the EI calculation.

Back Translation (BT) (Sennrich et al., 2016; Imamura et al., 2018). We utilize WMT'19 English-German translation models (Ng et al., 2019) based on Transformer (Vaswani et al., 2017) to translate the text from English to German,[4] and then translate it back to English. We use random sampling for decoding as recommended by (Xie et al., 2020; Edunov et al., 2018), and set the temperature to 0.8 to generate more diverse paraphrases.

Contextual Word Substitute (CWS) (Kobayashi, 2018). CWS masks some words in the original text stochastically, then predicts new words for substitution using a label-conditional language model (LM). The proportion α of words to be masked and predicted is 0.15.

Easy Data Augmentation (EDA) (Wei and Zou, 2019). Given a text from the training set, EDA randomly applies one of four simple editing operations including Synonym Replacement, Random Insertion, Random Swap, and Random Deletion. We set the proportion of words to be edited to 0.05 according to the recommendations in the paper.

Learning Data Manipulation (LDM) (Hu et al., 2019b). LDM is also a learnable data augmentation method, but uses only one operation: aforementioned CWS. LDM establishes a reinforcement learning system to iteratively optimize the parameters of the task model and label-conditional LM.

The main characteristics of the baselines are summarized in Table 1.

3.3 Experimental Settings

We choose large-scale pre-trained BERT (base, uncased version) (Devlin et al., 2019) as the backbone model. We adopt the Adam optimizer (Kingma and Ba, 2015) and a linear warmup scheduler with an initial learning rate of 4e-5. The training epoch is 20 for TREC and 10 for the reset datasets. We pick the best checkpoint according to the validation loss. Note that the validation loss is also the criterion for policy optimization, so we split out two validation sets with the same size during the phase of policy optimization. One is used for evaluating the model checkpoints, and the other is for evaluating the sampled policy. All experiments are conducted with 8 Tesla P40 GPUs. Following previous works (Hu et al., 2019b; Xie et al., 2020; Wei and Zou, 2019),

---

[4] https://github.com/pytorch/fairseq/blob/master/examples/translation/README.md

Baselines	Learnable	Compositional	Op. Level
BT	✗	✗	Sentence
CWS	✗	✗	Word
EDA	✗	✓	Word
LDM	✓	✗	Word
TAA (Ours)	✓	✓	Word

Table 1: Comparison of baselines.

we verify the performance of all data augmentation baselines in two special data scenarios:

Low-resource Regime. For every dataset in Section 3.1, we constrain the amount of available labeled data by sub-sampling a smaller training and validation set. In order to maintain the distribution of the original dataset, we apply *Stratified ShuffleSplit* (Esfahani and Dougherty, 2014) to split the training and validation set. The final datasets IMDB, SST-5, TREC, YELP-2, and YELP-5 have 80, 200, 120, 80, 200 labeled training samples, respectively, which pose significant challenges for learning a well-performing classifier. The number of validation samples is 60, 150, 60, 60, 150, respectively. In low-resource regime, we introduce a parameter n_{aug}, representing the magnification of augmentation. For example, $n_{\text{aug}} = 16$ means that we synthesize 16 samples for each given sample.

Class-imbalanced Regime. For binary sentiment classification datasets IMDB and SST-2, we sub-sample the training samples from an imbalanced class distribution. After sub-sampling, the negative class of the training set has 1000 samples, while the positive class has only 20/50/100 training samples respectively in three experiments. In this regime, we only augment the samples in positive class by 50/20/10 times, so that the final data amount of the positive and negative class are the same. We set up another baseline **Over-Sampling (OS)** for this scenario, which over-samples the training samples in the positive class by 50/20/10 times. Following the settings of the previous work (Ren et al., 2018; Hu et al., 2019b), the validation set for training and policy optimization is balanced.

In both low-resource and class-imbalanced regimes, the test set for final evaluation is **balanced** and **intact** without any reduction, which indicates that accuracy is a fair metric for evaluation. The size of policy N and the number of the operations sampled for each text N^* is 8 and 2, respectively. For a fair comparison, we use 3 random seeds for data sub-sampling and run 5 times under each seed.

Method	IMDB (80)	SST-5 (200)	TREC (120)	YELP-2 (80)	YELP-5 (200)	Average
No Aug	64.74±3.41	36.14±3.99	69.31±5.37	73.87±4.22	36.62±4.67	56.14±4.33
CWS	73.44±3.56	39.22±2.66	74.31±5.01	78.87±3.96	43.05±2.18	61.77±3.47
EDA	73.93±1.88	39.72±1.93	71.59±4.39	77.46±4.78	43.17±2.76	61.17±3.15
LDM	70.39±3.61	40.25±1.48	75.33±4.77	79.70±2.85	43.85±0.96	61.90±2.73
BT	74.13±3.10	39.40±3.87	78.08±4.07	78.97±2.95	42.19±2.53	62.55±3.30
TAA(Ours)	**75.68**[*]±3.27	**40.28**±1.80	**81.47**[**]±3.87	**81.75**[*]±3.57	**45.29**[*]±1.76	**64.89**±2.85

Table 2: Test accuracy (%) with standard deviation of different augmentation methods in low-resource regime. IMDB (80) means the number of training samples after sub-sampling is 80. [*] and [**] indicate statistically significant (p < .05 and p < .01) improvements over the best baseline. Here $n_{aug} = 16$.

Finally, we report the results on the average of 15 runs with standard deviation (denoted as ±).

3.4 Main Results

We train the model on the training set augmented by baselines in Section 3.2, then conduct evaluations on the balanced and intact test set.

Low-resource Regime. Table 2 shows the test accuracy of TAA on five datasets. In the low-resource regime, the model suffers from a severe overfitting problem. For example, the accuracy on the sub-training set of TREC is as high as 98.06%±1.34%, while only obtains 69.3%±5.4% on the test set. Several kinds of data augmentation baselines have greatly improved the generalization ability of the model. As shown in Figure 1, however, the performance of the augment operations is very sensitive to their parameters like the magnitude. The heuristic-based approaches such as EDA and CWS are likely to be trapped in sub-optimum because of the manually parameters setting. On the contrary, our Text AutoAugment algorithm with a learnable and compositional policy outperforms all the baselines by a considerable margin. Compared to the model without augmentation, TAA boosts the accuracy by **8.8%** averagely. Note that it does not cost too much time of computation to achieve such performance. When the number of iterations T (see Algorithm 1) is 200, it only requires 4.24±0.10 GPU hours on 8 NVIDIA Tesla P40 GPUs to finish the optimization.

Class-imbalanced Regime. To verify that TAA can also improve the performance of the model in class-imbalanced regime, we conduct experiments based on the settings in Section 3.3. As illustrated in Table 3, the over-sampling method alleviates the overfitting problem to some extent but is not as efficient as augmentation baselines. In contrast,
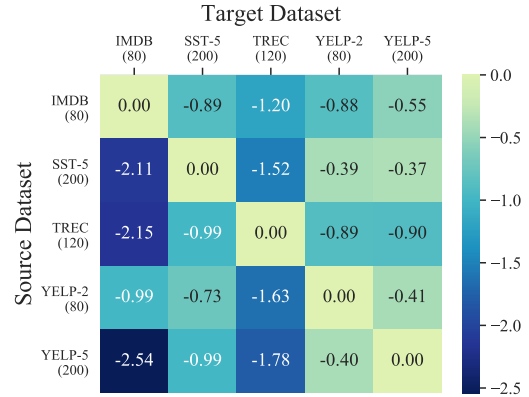

Figure 4: Transferability of the policies searched by TAA from one source dataset to other target datasets. The number denotes the performance degradation when the policy is transferred to the target dataset.

our TAA boosts the test accuracy by an average of about **9.7%**, which surpasses other algorithms.

In both low-resource and class-imbalanced regimes, our TAA framework based on *multiple learnable* operations surpasses BT & CWS based on *single unlearnable* operation, EDA based on *multiple unlearnable* operations, and LDM based on *single learnable* operation.

4 Analysis

We analyse the learned policies on four aspects including transferability, scalability, magnification and structure. Besides, we evaluate the diversity and semantics preservation of the augmented data, and further conduct a case study. The final policies searched by our TAA algorithm are listed in Appendix D.

4.1 Transferability

We conduct experiments to study whether the policies searched on one source dataset can be applied to other target datasets. As illustrated in Figure 4,

Method	SST-2 (20:1000)	SST-2 (50:1000)	SST-2 (100:1000)	IMDB (20:1000)	IMDB (50:1000)	IMDB (100:1000)	Average
No Aug	50.17±1.30	55.01±6.33	68.01±8.80	50.05±0.00	51.84±4.56	65.81±7.65	56.82±4.77
OS	52.01±1.80	58.58±5.56	68.97±3.48	52.29±2.69	59.08±6.33	68.63±4.67	59.93±4.09
BT	54.05±4.20	59.47±5.67	71.70±5.29	52.24±2.90	57.43±6.85	67.20±6.89	60.35±5.46
EDA	52.91±3.87	59.53±5.86	70.05±5.34	**57.29**±5.74	64.09±7.18	71.28±4.94	62.53±5.49
CWS	53.09±1.98	64.62±5.05	74.67±3.88	55.58±3.34	64.37±3.55	74.33±4.92	64.44±3.79
TAA(Ours)	**56.45***±3.67	**66.05**±4.85	**75.12**±5.20	56.92±2.80	**66.73***±4.20	**77.87***±3.18	**66.52**±3.98

Table 3: Test accuracy (%) with standard deviation in class-imbalanced regime. SST-2 (20:1000) means the positive class of SST-2 has 20 training samples while the negative class has 1000 training samples. * indicates statistically significant (p < .05) improvements over the best baseline.

Dataset	No Aug	TAA	Improvement ($\uparrow$)
IMDB	88.77	**89.37**	0.60
SST-5	52.29	**52.55**	0.26
TREC	96.40	**97.07**	0.67
YELP-2	95.85	**96.04**	0.19
YELP-5	65.55	**65.73**	0.18
Average	79.77	**80.15**	0.38

Table 4: Test accuracy (%) of TAA on full datasets. Here $n_{\text{aug}} = 4$ for IMDB, SST-5 and TREC. $n_{\text{aug}} = 2$ for YELP-2 and YELP-5.

Figure 5: Test accuracy (%) with different n_{aug} on SST-5 and TREC, respectively. Each curve in the sub-figure denotes a kind of baseline. Shaded regions indicate standard deviation over 15 trials.

the transfer of the policy to other datasets only results in a slight performance degeneration of 1.1% on average and can boosts the test accuracy by 7.7% comparing to no augmentation. The policies maintain better transferability when the gap of class number and sequence length[5] is smaller between the source and target datasets.

4.2 Scalability

In order to investigate whether the TAA policy is still helpful as the amount of training data increases, we apply the policies searched in the low-resource regime to the corresponding full datasets. Note that the only difference between the experimental setting here and that in the low-resource and class-imbalanced regimes is the amount of the training data. The test sets are intact and balanced in all three situations. As shown in Table 4, the policies searched on a subset with only 2k training samples on average have great scalability. Such policies can improve the quality and diversity of the full training set, and further, improve the model performance by an average of 0.38.
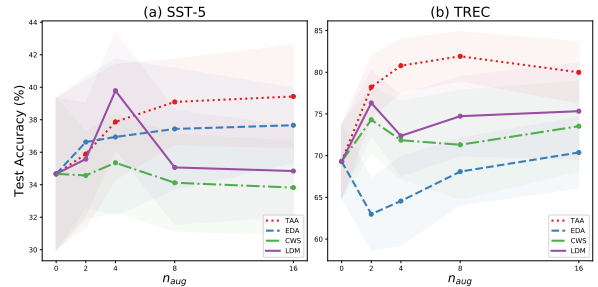
---

[5]Please refer to Appendix C

4.3 Impact of Augment Magnification

Figure 5 shows the results of each method with different augment magnification n_{aug}. We find that CWS and LDM achieve their best performance on SST-5 when $n_{\text{aug}} = 4$, but the accuracy drops sharply as n_{aug} continues to increase. The performance improvement from methods with a single operation cannot scale with more augmented samples, presumably because of the lack of novel information. Worsely, EDA, which is based on unlearnable multiple operations, even harms the model training on TREC, due to its inappropriate parameter setting with human experience (Wei and Zou, 2019). In virtue of the compositional and learnable policy, the augmented data synthesized by TAA are effective and perform well in most cases.

4.4 Impact of the Policy Structure

We change the size of the policy N and the number of the operations sampled for each text N^* and re-execute the experiments to explore the impact of the policy structure. The left panel in Figure 6 shows the results with different N^*. As N^* increases, the original text is likely to be applied with more operations sequentially, which causes slight
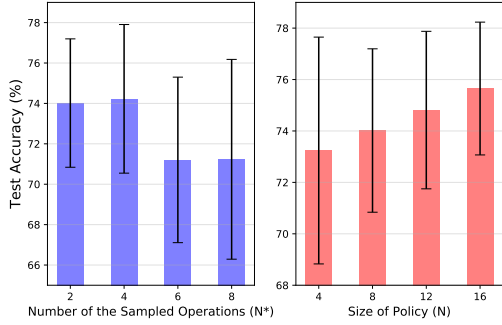
Figure 6: Performance of TAA with respect to the structure of augmentation policy.

Method	IMDB (80)	SST-5 (200)	TREC (120)	YELP-2 (80)	YELP-5 (200)	Average
CWS	0.302	0.731	0.827	0.390	0.393	0.529
EDA	0.304	0.747	0.839	0.397	0.399	0.537
BT	**0.345**	0.745	**0.849**	0.439	0.440	0.564
TAA(Ours)	**0.345**	**0.751**	0.841	**0.445**	**0.446**	**0.566**

Table 5: Distinct-2 of different augmentation methods in low-resource regimes.

damage to the performance of TAA. While the right panel shows that the more operations a policy contains, the better TAA performs. Therefore, for the compositional data augmentation algorithm, it is helpful to generate more diverse samples. Other ablation studies on operation type and searching algorithm can be found in Appendix E.

4.5 Diversity of Augmented Data

We evaluate the diversity of the augmented data by computing the Dist-2 (Li et al., 2016), which measures the number of distinct bi-grams in generated sentences. The metric is scaled using the total number of bi-grams in the generated sentence, which ranges from 0 to 1 where a larger value indicates higher diversity. As shown in Table 5, TAA achieves the best overall performance, even slightly outperforms the generation-based method, i.e., back-translation. The results validate the effectiveness of applying various editing operations in the compositional augmentation policy, and demonstrate the superiority of our well-designed algorithm for automatic hyper-parameters tuning.

4.6 Semantics Preservation of Augmented Data

The augmented sentence are supposed to preserve the semantic meaning of the original sentence, for enriching rather than deviating from the original data set support. We propose to evaluate semantic similarity between the generated sentences and

the original ones, based on sentence embedding cosine similarity. In more detail, for the sentence pair (x, x_{aug}) consists of the original sentence x and the corresponding augmented text x_{aug}, we utilize Sentence-BERT (Reimers and Gurevych, 2019) library, which achieves the state-of-the-art performance on various semantic textual similarity benchmarks, to obtain dense vector representations of sentences $(\mathbf{x}, \mathbf{x}_{aug})$. The semantic preservation score $\text{SP}(x, x_{aug})$ is defined as:

$$\text{SP}(x, x_{aug}) = \frac{\mathbf{x} \cdot \mathbf{x}_{aug}}{\|\mathbf{x}\| \|\mathbf{x}_{aug}\|}. \quad (6)$$

We compute the average semantic preservation score for the whole augmented datasets using different augmentation methods, and the results are listed in Table 6. It can be found that our TAA achieves the best results, comparable with the generation-based method back-translation. We note that our method achieves the highest semantic preservation score when the number of training samples is relatively small, which demonstrates that our method can generalize to extreme low-resource scenarios.

4.7 Case Study

Figure 7 shows some cases of augmented texts generated by TAA. Given the original text with the label "positive", the modifications applied by TAA distort the semantics and sentence structure at the beginning. As the number of iteration T increases, TAA captures the feature of the dataset adaptively, helping it achieve a better balance between diversity and quality on augmented samples.

5 Related Work

Previous data augmentation algorithms in NLP can be categorized into generation-based and editing-based methods. For generation-based methods, back-translation (Sennrich et al., 2016; Imamura et al., 2018; Luque, 2019; Zhang et al., 2020) generates the paraphrase of a text by translating it to an intermediate language and back. Kobayashi (2018) masks some words then uses a label-conditional language model to predict them. Anaby-Tavor et al. (2020) and Kumar et al. (2020) leverage various language models such as GPT-2 (Radford et al., 2019) and BART (Lewis et al., 2020) to generate a continuation of the original sentences. For editing-based methods, Wei and Zou (2019) propose the Easy Data Augmentation (EDA) technique, which randomly selects editing operations from four candidates for the augmentation. Xie et al. (2020)

Method	IMDB (80)	SST-5 (200)	TREC (120)	YELP-2 (80)	YELP-5 (200)	Average
CWS	0.698	0.796	0.784	0.739	0.736	0.751
EDA	0.738	0.665	0.610	0.785	**0.785**	0.717
BT	0.738	**0.799**	**0.892**	0.741	0.747	**0.783**
TAA(Ours)	**0.771**	0.776	0.847	**0.794**	0.726	**0.783**

Table 6: Cosine similarity of sentence embeddings between the augmented sentence and the original one. IMDB (80) means the number of training samples after sub-sampling is 80. Here $n_{\text{aug}} = 16$ and number of iterations T is 200.

Figure 7: Examples of augmented texts generated by TAA on SST-5 after 1/100/200 iteration of exploration. Red operation $\mathcal{O} = \langle t, p, \lambda \rangle$ indicates it is finally applied according to probability p. We use **RS** (Random Swap), **RD** (Random Delete), **TI** (TF-IDF Insert), **TS** (TF-IDF Substitute) and **WS** (WordNet Substitute) for abbreviation.

propose to substitute uninformative words with low TF-IDF scores. These methods require a lot of prior knowledge to preset their parameters and are prone to fall into the sub-optimum.

Recently, some algorithms are proposed for automatically learning augmentation policy in the field of Computer Vision (Cubuk et al., 2019; Ho et al., 2019; Lim et al., 2019; Hataya et al., 2020; Li et al., 2020) and NLP (Cai et al., 2020; Miao et al., 2021). However, their modelings are different from ours. Specifically, AutoAugment (Cubuk et al., 2019) establishes a Reinforcement Learning (RL) framework to search for the best augmentation policy. In order to reduce the time of policy exploration, Ho et al. (2019) replace the RL framework with a population-based algorithm. Lim et al. (2019) leverage 5-fold cross-validation and augments the validation set instead of the training set. While in our proposed approach, the objective function for policy optimization is designed as the loss that models achieve on the validation set after trained on the augmented set. Besides, we utilize SMBO as the optimization algorithm, which can return a promising result efficiently. In the field of NLP, TAA surpasses both EDA (Wei and Zou, 2019) and LDM (Hu et al., 2019b) via a learnable and compositional augmentation policy.

6 Conclusion

In this paper, we propose an effective method called Text AutoAugment (TAA) to establish a compositional and learnable paradigm for data augmentation. TAA regards a combination of various editing operations as an augmentation policy and utilizes SMBO for policy learning. Experiments show that TAA can substantially improve the generalization ability of models as well as lighten the burden of artificial augmentation designing.

Acknowledgements

References

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 *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7383–7390. AAAI Press.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7556–7566. Association for Computational Linguistics.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2546–2554.

Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. 2020. Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6334–6343. Association for Computational Linguistics.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021. An empirical survey of data augmentation for limited data learning in NLP. *CoRR*, abs/2106.07499.

Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. 2019. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 113–123. Computer Vision Foundation / IEEE.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 489–500. Association for Computational Linguistics.

Mohammad Shahrokh Esfahani and Edward R. Dougherty. 2014. Effect of separate sampling on classification accuracy. *Bioinform.*, 30(2):242–250.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2019. Auto-sklearn: Efficient and robust automated machine learning. In *Automated Machine Learning - Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 113–134. Springer.

Ryuichiro Hataya, Jan Zdenek, Kazuki Yoshizoe, and Hideki Nakayama. 2020. Faster autoaugment: Learning augmentation strategies using backpropagation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXV*, volume 12370 of *Lecture Notes in Computer Science*, pages 1–16. Springer.

Alex Hernández-García and Peter König. 2018. Data augmentation instead of explicit regularization. *CoRR*, abs/1806.03852.

Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2731–2741. PMLR.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019a. PARABANK: monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6521–6528. AAAI Press.

Zhiting Hu, Bowen Tan, Ruslan Salakhutdinov, Tom M. Mitchell, and Eric P. Xing. 2019b. Learning data manipulation for augmentation and weighting. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15738–15749.

Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, NMT@ACL 2018, Melbourne, Australia, July 20, 2018*, pages 55–63. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 452–457. Association for Computational Linguistics.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha P. Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3609–3619. Association for Computational Linguistics.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119. The Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy M. Hospedales, Neil Martin Robertson, and Yongxing Yang. 2020. DADA: differentiable automatic data augmentation. *CoRR*, abs/2003.03780.

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

Franco Martín Luque. 2019. Atalaya at TASS 2019: Data augmentation and robust embeddings for sentiment analysis. In *Proceedings of the Iberian Languages Evaluation Forum co-located with 35th Conference of the Spanish Society for Natural Language Processing, IberLEF@SEPLN 2019, Bilbao, Spain, September 24th, 2019*, volume 2421 of *CEUR Workshop Proceedings*, pages 561–570. CEUR-WS.org.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics.

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

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2786–2792. AAAI Press.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's WMT19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation, WMT 2019, Florence, Italy, August 1-2, 2019 - Volume 2: Shared Task Papers, Day 1*, pages 314–319. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 48–53. Association for Computational Linguistics.

Husam Quteineh, Spyridon Samothrakis, and Richard Sutcliffe. 2020. Textual data augmentation for efficient active learning on tiny datasets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7400–7410. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4331–4340. PMLR.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Patrice Y. Simard, Yann LeCun, John S. Denker, and Bernard Victorri. 1996. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 239–27. Springer.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 847–855. ACM.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

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

Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Quanming Yao, Mengshuo Wang, Hugo Jair Escalante, Isabelle Guyon, Yi-Qi Hu, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. 2018. Taking human out of learning applications: A survey on automated machine learning. *CoRR*, abs/1810.13306.

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

Yi Zhang, Tao Ge, and Xu Sun. 2020. Parallel data augmentation for formality style transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3221–3228. Association for Computational Linguistics.

A   Editing Operations for Augmentation

We use five simple and effect operations including Random Swap (Wei and Zou, 2019), Random Delete (Wei and Zou, 2019), WordNet Substitute (Zhang et al., 2015; Mueller and Thyagarajan, 2016; Wei and Zou, 2019), TF-IDF Substitute (Xie et al., 2020) and TF-IDF Insert (Xie et al., 2020) as the basic component of our augmentation policy. All operations used here are word-level, considering their low complexity and high effectiveness. The detailed description is shown in Table 7.

B   Sequential Model-based Global Optimization (SMBO)

Recall the TAA framework in Section 2, our objective is to search for the optimal augmentation policy $\mathcal{P}$ that minimizes the following loss:

$$\mathcal{J} = \mathcal{J}\left(f, \mathcal{D}_{\text{aug}}(\mathcal{P}), \mathcal{D}_{\text{val}}\right). \tag{7}$$

We leverage the Sequential Model-based Global Optimization (SMBO) as our optimizer for the policy learning. The optimization procedure is carried out in an iterative manner. To build the relation between the policy $\mathcal{P}$ and the objective loss $\mathcal{J}$ and sample the most promising policy at each iteration, we use the Tree-structured Parzen Estimator (TPE) (Bergstra et al., 2011) as a surrogate agent M to model the conditional probability $p_M(\mathcal{J} \mid \mathcal{P})$. Besides, we employ the following Expected Improvement (EI) criterion as an acquisition function for policy sampling in the current iteration:

$$\begin{aligned}
\text{EI}(\mathcal{P}) &= \mathbb{E}\left[\max\left(\mathcal{J}^\dagger - \mathcal{J}, 0\right)\right] \\
&= \int_{-\infty}^{\mathcal{J}^\dagger} \left(\mathcal{J}^\dagger - \mathcal{J}\right) p_M(\mathcal{J} \mid \mathcal{P}) d\mathcal{J}
\end{aligned} \tag{8}$$

Here, $\mathcal{J}^\dagger$ is a threshold and Eq. 8 stands for the expectation under the surrogate model that the loss $\mathcal{J}$ of a policy will exceed (negatively) the threshold $\mathcal{J}^\dagger$. We expect the loss of the sampled policy in each iteration to be smaller than the current threshold, thus the policy that maximizes the Expected Improvement will be chosen in the next iteration.

Instead of directly representing $p_M(\mathcal{J} \mid \mathcal{P})$ for calculating the Expected Improvement, the TPE builds a model of $p_M(\mathcal{P} \mid \mathcal{J})$ by applying Bayes rule. The TPE splits the historical observations of policies in two groups: the best performing one (e.g., the upper quartile) and the rest. The threshold $\mathcal{J}^\dagger$ is defined as the splitting value for the two

groups and the likelihood probability for being in each of these groups is modeled as:

$$p_M(\mathcal{P} \mid \mathcal{J}) = \begin{cases} l(\mathcal{P}) & \text{if } \mathcal{J} < \mathcal{J}^\dagger \\ g(\mathcal{P}) & \text{if } \mathcal{J} > \mathcal{J}^\dagger \end{cases} \tag{9}$$

Here, $l(\mathcal{P})$ models the distribution of previous sampled policies whose loss is less than the threshold, and $g(\mathcal{P})$ models the distribution of the other policies whose loss is greater than the threshold. The two densities l and g are modeled using Parzen estimators (also known as kernel density estimators), which are a simple average of kernels centered on existing data points. With Bayes Rule, we can prove that the Expected Improvement which we are trying to maximize is proportional to the ratio $l(\mathcal{P})/g(\mathcal{P})$:

$$\begin{aligned}
\text{EI}(\mathcal{P}) &= \int_{-\infty}^{\mathcal{J}^\dagger} \left(\mathcal{J}^\dagger - \mathcal{J}\right) p_M(\mathcal{J} \mid \mathcal{P}) d\mathcal{J} \\
&= \int_{-\infty}^{\mathcal{J}^\dagger} \left(\mathcal{J}^\dagger - \mathcal{J}\right) \frac{p_M(\mathcal{P} \mid \mathcal{J}) p(\mathcal{J})}{p(\mathcal{P})} d\mathcal{J} \\
&= \frac{l(\mathcal{P})}{p(\mathcal{P})} \int_{-\infty}^{\mathcal{J}^\dagger} \left(\mathcal{J}^\dagger - \mathcal{J}\right) p(\mathcal{J}) d\mathcal{J} \\
&= \frac{l(\mathcal{P})}{p(\mathcal{P})} \left[\gamma \mathcal{J}^\dagger - \int_{-\infty}^{\mathcal{J}^\dagger} p(\mathcal{J}) d\mathcal{J}\right] \\
&= \frac{l(\mathcal{P})}{\gamma l(\mathcal{P}) + (1-\gamma) g(\mathcal{P})} \left[\gamma \mathcal{J}^\dagger - \int_{-\infty}^{\mathcal{J}^\dagger} p(\mathcal{J}) d\mathcal{J}\right] \\
&\propto \left(\gamma + \frac{g(\mathcal{P})}{l(\mathcal{P})}(1-\gamma)\right)^{-1}
\end{aligned} \tag{10}$$

Accordingly, the maximization of the Expected Improvement can be achieved by maximizing the ratio $l(\mathcal{P})/g(\mathcal{P})$. In other words, we should sample the polices which are more likely under $l(\mathcal{P})$ than under $g(\mathcal{P})$. The TPE works by drawing sample polices from $l(\mathcal{P})$, evaluating them in terms of $l(\mathcal{P})/g(\mathcal{P})$, and returning the set that yields the highest value under $l(\mathcal{P})/g(\mathcal{P})$ corresponding to the greatest expected improvement. These policies are then evaluated on the objective function and the results are merged into the observation history. The algorithm builds $l(\mathcal{P})$ and $g(\mathcal{P})$ using the history to update the probability model M of the objective function that improves with each iteration.

C   Statistics of Datasets

We conduct experiments on six popular datasets including IMDB (Maas et al., 2011), SST-2, SST-5 (Socher et al., 2013), TREC (Li and Roth, 2002), YELP-2 and YELP-5 (Zhang et al., 2015). The statistics of datasets used are listed in Table 8.

Operation Name	Description
Random Swap (RS)	Swap two adjacent words randomly.
Random Delete (RD)	Delete words randomly.
WordNet Substitute (WS)	Substitute words with their synonyms according to WordNet.
TF-IDF Substitute (TS)	Substitute uninformative words with low TF-IDF scores.
TF-IDF Insert (TI)	Insert informative words with high TF-IDF.

Table 7: Description of all the augment operations in the search space.

Dataset	# Classes	# Train samples	# Test samples	Average length	Task
IMDB	2	25,000	25,000	326	Sentiment Analysis
SST-2	2	7,791	1,821	18	Sentiment Analysis
SST-5	5	9,643	2,210	19	Sentiment Analysis
TREC	6	5,452	500	10	Question Classification
YELP-2	2	560,000	38,000	139	Review Classification
YELP-5	5	650,000	50,000	141	Review Classification

Table 8: Statistics on the datasets.

Policy	IMDB (80)	SST-5 (200)	TREC (120)	YELP-2 (80)	YELP-5 (200)
Op1	$\langle TS, 0.77, 0.07 \rangle$	$\langle RD, 0.44, 0.22 \rangle$	$\langle WS, 0.75, 0.44 \rangle$	$\langle TS, 0.95, 0.26 \rangle$	$\langle RD, 0.34, 0.32 \rangle$
Op2	$\langle TS, 0.50, 0.31 \rangle$	$\langle WS, 0.59, 0.50 \rangle$	$\langle WS, 0.33, 0.26 \rangle$	$\langle RD, 0.08, 0.49 \rangle$	$\langle WS, 0.69, 0.19 \rangle$
Op3	$\langle RD, 0.72, 0.05 \rangle$	$\langle TI, 0.66, 0.11 \rangle$	$\langle RS, 0.60, 0.02 \rangle$	$\langle WS, 0.57, 0.41 \rangle$	$\langle RD, 0.70, 0.15 \rangle$
Op4	$\langle TI, 0.66, 0.09 \rangle$	$\langle TS, 0.11, 0.25 \rangle$	$\langle WS, 0.59, 0.36 \rangle$	$\langle RD, 0.85, 0.02 \rangle$	$\langle RD, 0.85, 0.27 \rangle$
Op5	$\langle TI, 0.69, 0.13 \rangle$	$\langle WS, 0.69, 0.09 \rangle$	$\langle TI, 0.59, 0.17 \rangle$	$\langle RS, 0.30, 0.15 \rangle$	$\langle TS, 0.72, 0.35 \rangle$
Op6	$\langle RS, 0.26, 0.05 \rangle$	$\langle WS, 0.41, 0.07 \rangle$	$\langle WS, 0.63, 0.34 \rangle$	$\langle TS, 0.55, 0.05 \rangle$	$\langle RD, 0.58, 0.47 \rangle$
Op7	$\langle TS, 0.77, 0.50 \rangle$	$\langle TS, 0.69, 0.02 \rangle$	$\langle TS, 0.22, 0.37 \rangle$	$\langle TS, 0.63, 0.34 \rangle$	$\langle RS, 0.48, 0.22 \rangle$
Op8	$\langle TS, 0.36, 0.25 \rangle$	$\langle WS, 0.99, 0.22 \rangle$	$\langle RS, 0.96, 0.03 \rangle$	$\langle WS, 0.13, 0.28 \rangle$	$\langle TS, 0.72, 0.50 \rangle$

Table 9: TAA policies searched in the low-resource regime. Each policy finally consists of 8 atomic editing operations and each operation satisfies the form of $\mathcal{O} = \langle t, p, \lambda \rangle$. We use **RS** (Random Swap), **RD** (Random Delete), **WS** (WordNet Substitute), **TI** (TF-IDF Insert), **TS** (TF-IDF Substitute) for simplification.

D   Searched Policy

The policies searched by our TAA algorithm are listed in Table 9. Each policy finally consists of 8 atomic editing operations and each operation satisfies the form of $\mathcal{O} = \langle t, p, \lambda \rangle$. All the policies can be used directly to augment the full training set to further boost the model performance on the corresponding downstream tasks.

E   Ablation Study of TAA Policy

E.1   Searching Algorithm Ablation

To verify the effectiveness of our optimization algorithm, we re-execute the experiment on the same search space described in Section 2.1, but do NOT conduct any policy optimization. For each text in the training set, a random policy is sampled to synthesize the augmented data. We call this method Text AutoAugment-Random (**TAA-R**). Table 10

illustrates the results of TAA and TAA-R. Generally, the augmentation policy after optimization performs better than the random policy by 1.92%. Note that we incorporate prior knowledge in Section 2.1 to constrain the range of the operation magnitude, which ensures the performance of the operation and avoids generating bad samples.

E.2   Operation Type Ablation

We conduct operation type ablation study to examine the effect of the operation types search space. Specifically, we eliminate one operation type while keeps others for searching the optimal policy, and evaluate the task performance using the learned policy. The task performance difference with different ablated operations on IMDB and SST-2 dataset are shown in Table 11. We find that the elimination of operation type generally leads to a decrease of task performance, indicating that the effect of differ-

Method	Low-resource Regime			Class-imbalanced Regime			Overall
	IMDB (80)	SST-5 (200)	TREC (120)	IMDB (20:1000)	IMDB (50:1000)	IMDB (100:1000)	Average
TAA-R	72.64±5.55	38.60±2.12	77.72±4.01	55.72±3.74	63.60±5.71	74.37±4.22	63.78±4.23
TAA	**75.68***±3.27	**40.28****±1.80	**81.47****±3.87	**56.92**±2.80	**66.73***±4.20	**77.87***±3.18	**66.49**±3.19

Table 10: Test accuracy (%) for TAA-R and TAA in low-resource and class-imbalanced regime. * and ** indicate statistically significant (p < .05 and p < .01) improvements over TAA-R.

Dataset	Full Space	w/o RS	w/o RD	w/o TI	w/o TS	w/o WS
IMDB(80)	75.68	−0.12	−1.03	+0.89	−0.07	−1.47
SST-2(50:1000)	66.05	−2.09	−1.64	−0.94	−0.47	+1.27

Table 11: Test accuracy (%) in the ablation study of operation type. We use **RS** (Random Swap), **RD** (Random Delete), **WS** (WordNet Substitute), **TI** (TF-IDF Insert), **TS** (TF-IDF Substitute) for simplification.

ent operations are complementary with each other. We attribute it to that more operations types will improve the diversity of the combinations of text manipulation, thus boosting the dataset quality and benefiting the generalizability of the model.