Adversarial Word Dilution as Text Data Augmentation in Low-Resource Regime

Junfan Chen[1], Richong Zhang[1,2*], Zheyan Luo[1], Chunming Hu[1,2], Yongyi Mao[3]

[1]SKLSDE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada
chenjf@act.buaa.edu.cn, zhangrc@act.buaa.edu.cn, akamya@buaa.edu.cn, hucm@buaa.edu.cn, ymao@uottawa.ca

Abstract

Data augmentation is widely used in text classification, especially in the low-resource regime where a few examples for each class are available during training. Despite the success, generating data augmentations as hard positive examples that may increase their effectiveness is under-explored. This paper proposes an Adversarial Word Dilution (AWD) method that can generate hard positive examples as text data augmentations to train the low-resource text classification model efficiently. Our idea of augmenting the text data is to dilute the embedding of strong positive words by weighted mixing with unknown-word embedding, making the augmented inputs hard to be recognized as positive by the classification model. We adversarially learn the dilution weights through a constrained min-max optimization process with the guidance of the labels. Empirical studies on three benchmark datasets show that AWD can generate more effective data augmentations and outperform the state-of-the-art text data augmentation methods. The additional analysis demonstrates that the data augmentations generated by AWD are interpretable and can flexibly extend to new examples without further training.

Introduction

Training effective text classification models often relies on sufficient precisely labeled data. However, it is common in some real-world scenarios that collecting plenty of valid text data is difficult, e.g., requiring considerable effort from human annotators. Enabling the model to learn efficiently with limited resources therefore becomes a practical need and hotspot in the industry and research communities. For example, text classification in few-shot learning that trains on small tasks composed of a few examples and tests on new tasks of unseen classes (Yu et al. 2018; Geng et al. 2019; Chen et al. 2022), semi-supervised learning that provides a few labeled texts and plenty of unlabeled texts for training (Miyato, Dai, and Goodfellow 2017; Xie et al. 2020; Lee, Ko, and Han 2021) and the low-resource regime that also provides a few labeled texts in training but no unlabeled texts available (Wei and Zou 2019; Wu et al. 2022). Data augmentation is widely used in these tasks to increase data size and boost training and is often evaluated in the low-resource regime, which we will focus on in this paper.

*Corresponding author: zhangrc@act.buaa.edu.cn

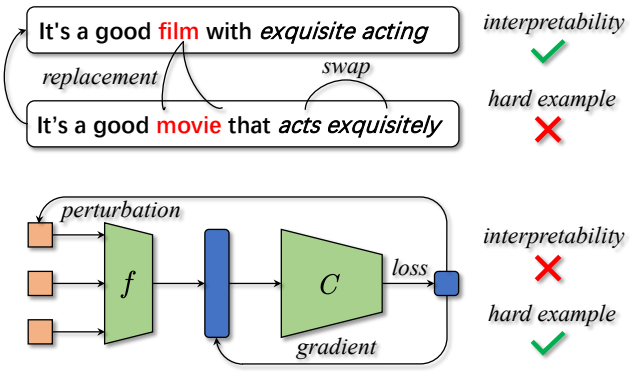


Figure 1: Illustration of contextual augmentation and representational augmentation (taking adversarial data augmentation as an example). The upper and lower part of the figure respectively show an example of contextual augmentation and the process of adversarial data augmentation. Contextual augmentation is interpretable but may not generate hard positive examples. Adversarial data augmentation can generate hard positive examples but is short of interpretability.

Text data augmentation is challenged by the discrete nature and complex semantics of texts. Despite the challenges, two branches of textual data augmentation techniques have been explored and demonstrated effective: *contextual augmentation* and *representational augmentation*. The contextual augmentation methods augment the texts by replacing, inserting, deleting or swapping words (Kolomiyets, Bethard, and Moens 2011; Wang and Yang 2015; Artetxe et al. 2018; Gao et al. 2019; Wei and Zou 2019; Miao et al. 2020), or paraphrasing the original texts (Edunov et al. 2018; Chen et al. 2019; Kumar et al. 2019; Dopierre, Gravier, and Logerais 2021). For example, in the upper part of Figure 1, given the text "*It's a good movie that acts exquisitely*," one of its contextual augmentations may be "*It's a good film with exquisite acting*." The contextual augmentations are semantically interpretable because the modifications are expected to keep semantic consistency with the original texts. However, such heuristic modifications may produce simple data augmentations that bring little improvements or even degraded performance (Zhang, Zhao, and LeCun 2015).

The representational augmentation methods generate augmented inputs by interpolation or perturbation of the word embedding or text representations (Miyato, Dai, and Goodfellow 2017; Hsu, Tang, and Glass 2018; Wu et al. 2019b; Chen et al. 2020; Jiang et al. 2020; Zhu et al. 2020; Chen et al. 2021). One most intensively used perturbation method is adversarial data augmentation, which creates augmented examples by adding adversarial noise to the original representations (Miyato, Dai, and Goodfellow 2017; Jiang et al. 2020; Zhu et al. 2020). As shown in the lower part of Figure 1, since the adversarial noise is generated by gradients minimizing the model objective, the adversarial data augmentations could be regarded as hard positive examples. However, the augmentations lack semantic interpretability because the noise is semantic-independent and the perturbed representations no longer represent any valid word. Moreover, adversarial perturbation requires gradient computation for each example, making it inflexible to extend to new examples.

Hard positive examples have been proven to be effective in improving model robustness (Schroff, Kalenichenko, and Philbin 2015; Khosla et al. 2020). Thus, we hope to develop a data augmentation method that can produce hard positive examples without losing interpretability and extensibility. Based on this objective, we propose to generate hard data augmentations by weighted mixing the embedding of strong positive words with unknown-word embedding. Our motivation is to dilute the strong positive words, making the texts become more neutral and turn into hard positive examples. For example, the sentence "*It's a good movie that acts exquisitely*" expresses the *positive* sentiment supported by strong positive words *good* and *exquisitely*. Diluting their expressiveness makes the sentence become harder because the sentence semantics becomes less positive. Our Word dilution method is a practical realization since it is easy to implement and well interpretable. The remaining challenge is how to acquire the dilution weight assigned to each word.

To not struggle to estimate the dilution weights manually or heuristically, motivated by Generative Adversarial Networks (Goodfellow et al. 2014), we automatically learn the dilution weights and train the classifier by the constrained adversarial optimization process. Specifically, we introduce neural networks (i.e., dilution networks) to produce the dilution weights. At the inner-loop step, we fix the classifier and learn the dilution weights by maximizing the loss. At the outer-loop step, we fix the dilution weights and train the classifier by minimizing the loss with augmented inputs. We also use separate dilution networks for different classes to guide the dilution-weight learning process with the label information. As the dilution networks are learned independent of the classifier, they can be extended to compute dilution weights for new examples without further training.

To summarize, our work makes the following contributions. (1) We propose AWD data augmentation that generates hard positive examples by diluting strong positive words with unknown-word embedding. (2) We adversarially learn the dilution weights by the constrained min-max optimization with the guidance of the labels. (3) We empirically show that AWD outperforms the state-of-the-art data augmentations and demonstrates its interpretability and extensibility.

Related Works

Low-resource text classification aims to learn a classification model from a few labeled examples. The earlier work EDA (Wei and Zou 2019) generates text augmentations by random replacement, insertion, swap, deletion of words and trains the model on a small fraction of the original training data. The recent work explores manipulating the text data by augmenting and weighting data examples (Hu et al. 2019). Another recent work studies text data augmentations using different pre-trained transformer models (Kumar, Choudhary, and Cho 2020). The text smoothing model converts the text data from one-hot representations to controllable smoothed representations (Wu et al. 2022). Following these works, we also evaluate our text augmentation method AWD in low-resource text classification.

Text data augmentation techniques can be divided into contextual augmentation approaches and representational augmentation approaches. Besides EDA, contextual augmentation approaches also include works that substitute words using synonyms (Kolomiyets, Bethard, and Moens 2011; Wang and Yang 2015; Miao et al. 2020), randomly delete, insert, replace and swap words (Iyyer et al. 2015; Xie et al. 2017; Artetxe et al. 2018), replace or re-combine fragments of the sentences (Jia and Liang 2016; Andreas 2020), paraphrase the original text (Edunov et al. 2018; Chen et al. 2019; Kumar et al. 2019; Dopierre, Gravier, and Logerais 2021), replace or generate words using language models (Fadaee, Bisazza, and Monz 2017; Kobayashi 2018; Anaby-Tavor et al. 2020; Yang et al. 2020).

The representational augmentation approaches generate augmented inputs by interpolation or perturbation of the representations, i.e., word embedding or text representations. One popular interpolation-based representational augmentation method is Mixup (Zhang et al. 2018) which generates augmented examples by linear interpolations of the pair of data points and labels. This method has recently been intensively used in NLP tasks (Guo, Kim, and Rush 2020; Zhang, Yu, and Zhang 2020). For example, some works interpolate the original data with word-level augmented data (Miao et al. 2020) or adversarial data (Cheng et al. 2020), or interpolate the hidden representations of the original texts and augmented texts (Chen, Yang, and Yang 2020). Adversarial perturbation is often used to generate perturbation-based text augmentations, e.g., applying adversarial perturbations to the word embeddings (Miyato, Dai, and Goodfellow 2017; Zhu et al. 2020; Chen et al. 2020, 2021) or sentence representations (Hsu, Tang, and Glass 2018; Wu et al. 2019b).

Improving the interpretability of adversarial perturbation methods is challenging. Recent works in this direction guide word modification by adversarial training instead of directly applying perturbation to the representations (Ren et al. 2019; Cheng, Jiang, and Macherey 2019; Garg and Ramakrishnan 2020). Although these methods can produce interpretable examples, the generated data augmentations sacrifice some adversarial nature by replacing the optimal adversarial representations with approximated word embeddings. They also need adversarial modification on each example, making them inflexible to extend to new examples.

Problem Formulation

Text classification in the low-resource regime adopts supervised learning but is restricted by the available number of training examples. This task aims to efficiently learn a classification model given only a few labeled texts.

Formally, in low-resource text classification, let $\mathcal{Y}$ be the set of all classes of interest. For each class $y \in \mathcal{Y}$, we are only given a small number of k labeled texts for training. Let $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_{k\times|\mathcal{Y}|}, y_{k\times|\mathcal{Y}|})\}$ be the training set which consists of $k \times |\mathcal{Y}|$ labeled texts. Each (x_i, y_i) pair in $\mathcal{D}$ denotes a text x_i and its label y_i. Given the training set $\mathcal{D}$, the objective is to obtain a sufficiently robust model under the limited resource.

As the main challenge in this task is to deal with the overfitting problem caused by a few training examples, the previous works focus on designing various data augmentations as supplementary data to improve the model's robustness. In this paper, we follow this direction and concentrate our topic on text data augmentations in the low-resource regime.

Methodology

In this section, we first make a general illusion of our word dilution method and the challenges in dilution weights acquisition, then introduce our systematic solution that learns the dilution weights with neural networks and trains with the text classifier by adversarial min-max optimization.

General Illusion of Word Dilution

To generate hard positive examples as text data augmentations while keeping their semantic interpretability, we need to make word modification dependent on their semantics and simultaneously let the modified text hard to be recognized as positive by the text classifier. Considering that the polarity of a text is significantly determined by some emblematic words. As shown from the example in Figure 2, the emblematic words "*good*" and "*exquisitely*" indicate that the sentence "*It's a good movie that acts exquisitely*" expresses the positive sentiment. We call such emblematic words *strong positive words*. When we weaken the expressiveness of these strong positive words, the semantics of the text becomes more neutral and harder to be recognized by the classifier. This nature of the text data motivates us to design a new text data augmentation strategy *word dilution* shown in Figure 2 that dilutes the expressiveness of strong positive words in the original sentence by weighted mixing their embeddings with unknown word embedding, allowing us to generate hard positive examples without losing interpretability.

Word Dilution For an input text example $x_i \in \mathcal{D}$, let the sequence $\{w_{i1}, w_{i2}, \cdots, w_{in_i}\}$ be the set of all words that consist of the text, where n_i denotes the total number of words in x_i. For each word w_{ij} in x_i, we can obtain its word embedding $\mathbf{w}_{ij} \in \mathbb{R}^d$ by an embedding function e. We define the embedding function and obtain $\mathbf{w}_{ij}$ as follows

$$\mathbf{w}_{ij} = e(w_{ij}; \mathbf{E}), \tag{1}$$

where $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the parameters of the embedding function e, i.e., the embedding matrix, which is learnable during training. And $|\mathcal{V}|$ denotes the vocabulary size.

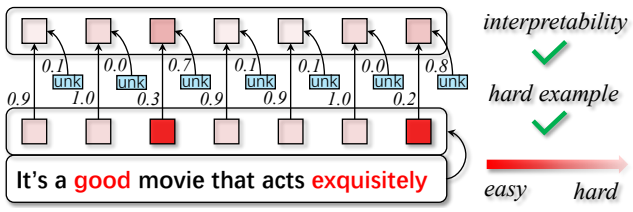


Figure 2: General illusion of word dilution. The squares represent word embeddings. The deeper color manifests that the word has more positive polarity. The weights beside the arrows represent weights of input words or the unknown word.

In a neural classification model, the word-embedding sequence $\{\mathbf{w}_{i1}, \mathbf{w}_{i2}, \cdots, \mathbf{w}_{in_i}\}$ of the input text is sent to the classifier and makes predictions. Word dilution intends to create hard positive examples challenging the classifier by weighted mixing the input word embeddings with unknown-word embedding. Specifically, we use the mark unk to denote the unknown word, whose embedding can also be obtained with the function e. We assign a dilution weight $\alpha_{ij} \in [0, 1]$ for each word w_{ij} according to its relevance to the text polarity and compute the diluted word $\tilde{\mathbf{w}}_{ij}$ by

$$\tilde{\mathbf{w}}_{ij} = (1 - \alpha_{ij})e(w_{ij}; \mathbf{E}) + \alpha_{ij}e(\mathrm{unk}; \mathbf{E}) \tag{2}$$

After getting all diluted word embeddings $\tilde{\mathbf{w}}_{ij}$, we can treat the sequence $\{\tilde{\mathbf{w}}_{i1}, \tilde{\mathbf{w}}_{i2}, \cdots, \tilde{\mathbf{w}}_{in_i}\}$ (simplified as $\{\tilde{\mathbf{w}}_{ij}\}$) as the data augmentation of the original input word-embedding sequence to further train the text classification model.

Challenges in Dilution-Weight Acquisition The acquisition of dilution weights suffers two challenges. (1) The dilution weights can be obtained from experience or heuristic methods, e.g., manually selecting the strong positive words for each text or heuristically finding them according to the semantic relevance of the word embeddings. However, these methods are usually laborious and inflexible to extend to new examples. (2) An appropriate α_{ij} is essential to guarantee the effect of word dilution, but it is difficult to decide. We expect the high α_{ij} for a word closely related to the polarity of the text, i.e., the strong positive word; so that the expressiveness of these words is diluted significantly according to Equation (2). However, this does not mean that assigning a highest α_{ij} for each strong positive word is the best because an extremely high α_{ij} may completely obliterate the semantics of the strong positive word and make the input text meaningless or even become a false positive example. Such meaningless data augmentations will no longer be considered as positive examples and may harm the model training. To tackle the challenges, our solution is to adversarially learn the dilution weights with neural networks through a constrained min-max optimization process.

Systematic Solution: Adversarial Word Dilution

The optimization of our adversarial word dilution (AWD) is shown in Figure 3, which consists of training the classifier with original inputs and adversarially learning the dilution weights and optimizing the classifier with augmented inputs.

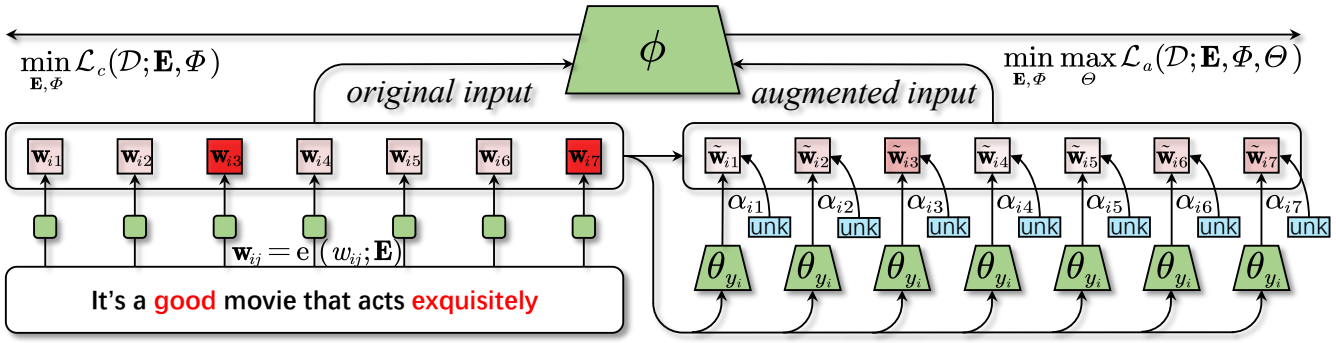


Figure 3: Adversarial word dilution. The ϕ and θ_{y_i} blocks respectively denote the text classifier and dilution networks.

Text Classifier The pre-trained language models, such as BERT (Devlin et al. 2019), have recently been employed in text classification and achieved promising results. Following previous works (Hu et al. 2019; Kumar, Choudhary, and Cho 2020; Wu et al. 2022) in low-resource text classification, we also choose BERT as the text classifier. Specifically, BERT takes the sequence of word embeddings $\{\mathbf{w}_{i1}, \mathbf{w}_{i2}, \cdots, \mathbf{w}_{in_i}\}$ (simplified as $\{\mathbf{w}_{ij}\}$) as inputs and outputs a score corresponding to each class y, which we denote as $s(\{\mathbf{w}_{ij}\}, \phi_y)$, where ϕ_y denotes the parameters corresponding to y. The class y with the highest score will be predicted as the label of a given text. With training set $\mathcal{D}$, we can optimize BERT by the following cross-entropy loss

$$\begin{aligned}\mathcal{L}_c(\mathcal{D};\mathbf{E},\Phi) &= \sum_{i=1}^{k\times|\mathcal{Y}|} \ell_c(x_i, y_i; \mathbf{E}, \Phi) \\ &= -\sum_{i=1}^{k\times|\mathcal{Y}|} \log \frac{\exp(s(\{\mathbf{w}_{ij}\}, \phi_{y_i}))}{\sum_{y\in\mathcal{Y}} \exp(s(\{\mathbf{w}_{ij}\}, \phi_y))},\end{aligned} \tag{3}$$

where $\ell(x_i, y_i)$ is the loss corresponds to the i^{th} example in $\mathcal{D}$ and Φ denotes the set of parameters $\{\phi_y : y \in \mathcal{Y}\}$.

Dilution Networks To produce text data augmentations through word dilution, we propose the label-guided dilution networks to obtain the dilution weights $\{\alpha_{i1}, \alpha_{i2}, \cdots, \alpha_{in_i}\}$ (simplified as $\{\alpha_{ij}\}$). Our solution is two-fold: on the one hand, to prevent obtaining the dilution weights by laborious experience or heuristic methods as well as enable the data augmentations to flexibly extend to new examples, we attempt to learn the dilution weights using neural networks; on the other hand, to allow the dilution weights to be learned in accordance with the semantics of different classes, we use separate neural networks for each class and train the dilution weights guided by the labels. Specifically, for each $y \in \mathcal{Y}$, we use a Multilayer Perceptron (MLP) parameterized by θ_y to produce the dilution weights for all words in the input text, i.e., the dilution weight of w_{ij} is computed by

$$\alpha_{ij} = \sigma(\theta_{y_i}^{\mathrm{T}}\, \mathrm{e}(w_{ij}; \mathbf{E}) + b_{y_i}), \tag{4}$$

where σ is the logistic function and b_{y_i} is bias. For convenience, we denote all parameters $\{\theta_y, b_y : y \in \mathcal{Y}\}$ as Θ.

Adversarial Optimization We leverage the min-max optimization derived form GAN (Goodfellow et al. 2014) to adversarially update the dilution weights and train the text classifier with augmented inputs. To that end, we first modify the classification loss in Equation (3) to include the dilution weights $\{\alpha_{ij}\}$ and augmented inputs $\{\tilde{\mathbf{w}}_{ij}\}$ as follows

$$\begin{aligned}\mathcal{L}_a(\mathcal{D};\mathbf{E},\Phi,\Theta) &= \sum_{i=1}^{k\times|\mathcal{Y}|} \ell_a(x_i, y_i, \{\alpha_{ij}\}; \mathbf{E}, \Phi, \Theta) \\ &= -\sum_{i=1}^{k\times|\mathcal{Y}|} \log \frac{\exp(s(\{\tilde{\mathbf{w}}_{ij}\}, \phi_{y_i}))}{\sum_{y\in\mathcal{Y}} \exp(s(\{\tilde{\mathbf{w}}_{ij}\}, \phi_y))}\end{aligned} \tag{5}$$

To learn appropriate dilution weights, we expect not to generate extreme $\{\alpha_{ij}\}$. Thus, for each i, we optimize the dilution networks under the following condition

$$\mathcal{R}_i(\{\alpha_{ij}\}; \Theta) = \|\{\alpha_{ij}\}\|_1 - \rho n_i \leqslant 0 \tag{6}$$

where $\|\{\alpha_{ij}\}\|_1 = \sum_{j=1}^{n_i} |\alpha_{ij}|$ and $\rho \in (0, 1)$ controls the amount of dilution allowed. This constraint guarantees that no more than ρ fraction of words are diluted.

Based on Equations (5) and (6), we have the following constrained min-max optimization problem

$$\min_{\mathbf{E},\Phi} \max_{\Theta} \mathcal{L}_a(\mathcal{D}; \mathbf{E}, \Phi, \Theta) \;\; s.t. \sum_{i=1}^{k\times|\mathcal{Y}|} \mathcal{R}_i(\{\alpha_{ij}\}; \Theta) \leqslant 0 \tag{7}$$

Note that the inner-loop constrained maximization problem can be converted to an unconstrained problem via Penalty Function method, then the optimization problem turns to

$$\min_{\mathbf{E},\Phi} \max_{\Theta} [\mathcal{L}_a(\mathcal{D};\mathbf{E},\Phi,\Theta) - \lambda \sum_{i=1}^{k\times|\mathcal{Y}|} \max(\mathcal{R}_i(\{\alpha_{ij}\}; \Theta), 0)], \tag{8}$$

where $\lambda \geqslant 0$ is the weight of the constraint item.

It is easy to control the dilution-weight range by modifying ρ. However, as $\|\{\alpha_{ij}\}\|_1 \leqslant \rho n_i$ is a strict constraint, it may force the model to produce undesirable lower dilution weights for all words. To encourage generating some high dilution weights, we loose the strict constraint as

$$\min_{\mathbf{E},\Phi} \max_{\Theta} [\mathcal{L}_a(\mathcal{D};\mathbf{E},\Phi,\Theta) - \gamma \sum_{i=1}^{k\times|\mathcal{Y}|} \|\{\alpha_{ij}\}\|_1 \frac{1}{n_i}]. \tag{9}$$

Algorithm 1: Training the Classification Model with AWD

Input: training set $\mathcal{D}$ and parameters $\mathbf{E}$, Φ, Θ
Output: the optimized dilution networks and classifier

initialize $\mathbf{E}$, Φ with BERT, randomly initialize Θ;
repeat
  for each (x_i, y_i) in $\mathcal{D}$ **do**
    obtain $\{\mathbf{w}_{ij}\}$ through $e(w_{ij}; \mathbf{E})$;
  end for
  input all $\{\mathbf{w}_{ij}\}$, update $\mathbf{E}$, Φ by $\min \mathcal{L}_c(\mathcal{D}; \mathbf{E}, \Phi)$;
  fix $\mathbf{E}$, Φ, update Θ by $\max \mathcal{L}_a(\mathcal{D}; \mathbf{E}, \Phi, \Theta)$;
  compute all $\{\alpha_{ij}\}$, generate all $\{\tilde{\mathbf{w}}_{ij}\}$ by eq. (2);
  fix Θ, update $\mathbf{E}$, Φ by $\min \mathcal{L}_a(\mathcal{D}; \mathbf{E}, \Phi, \Theta)$;
until convergence

Training Text Classification Model with AWD

We train the text classification model with adversarial word dilution by iteratively executing three optimization steps shown in Algorithm 1 (to describe the process easier, we omit the constraint items in (8) and (9)): (1) input the original data and minimize $\mathcal{L}_c(\mathcal{D}; \mathbf{E}, \Phi)$; (2) fix $\mathbf{E}$, Φ and update Θ by maximizing $\mathcal{L}_a(\mathcal{D}; \mathbf{E}, \Phi, \Theta)$, compute dilution weights and generate augmented data; (3) fix Θ, input the augmented data and update $\mathbf{E}$, Φ by minimizing $\mathcal{L}_a(\mathcal{D}; \mathbf{E}, \Phi, \Theta)$. We perform one SGD updating at each optimization step.

For convenience, we name the model optimized with the strict constraint term in (8) as **AWD(strict)** and loosed constraint item in (9) as **AWD** or **AWD(loose)**, respectively.

Experiment

Datasets and Experiment Setting

Following the previous works in low-resource text calssification (Wei and Zou 2019; Hu et al. 2019; Kumar, Choudhary, and Cho 2020; Wu et al. 2022), we evaluate our data augmentations on three benchmarks: SST, TREC, and SNIPS.

SST-2 Stanford Sentiment Treebank (SST) (Socher et al. 2013) is a sentiment classification dataset. The text data are extracted from movie reviews of the rottentomatoes.com website and is originally collected by (Pang and Lee 2005). The texts involve 2 classes, i.e., positive and negative.

TREC TREC (Li and Roth 2002) is a fine-grained question classification dataset. The text data is collected from USC, TREC 8, TREC 9 and TREC 10 questions. The 500 questions from TREC 10 are used for test. This dataset contains 6 question types (including person, location, etc.).

SNIPS SNIPS (Coucke et al. 2018) is an English dataset for natural language understanding, which is widely used in intent classification. The text data is collected from crowdsourced queries. This dataset contains 7 user intents from different domains (such as movie, book, etc.).

Experiment Setting We use the datasets provided by (Wu et al. 2022). To simulate low-resource text classification, we randomly select $k = 10, 20, 50$ examples for each class as the training sets. The training set with $k = 10$, the validation set and the test set are the same in (Wu et al. 2022). The data statistics are shown in Table 1.

Dataset	#train	#low-res	#val	#test	#class
SST	6,228	10/20/50	20	1821	2
TREC	5,406	10/20/50	60	500	6
SNIPS	13,084	10/20/50	70	700	7

Table 1: The statistics of the datasets. #low-res denotes the training examples per class (i.e., k) in low-resource regime.

Baseline Models and Implementation

Baseline Models We compare our AWD with the following baseline models: the pure **BERT** (Devlin et al. 2019) without data augmentations; contextual augmentation by word modification or paraphrasing, including **EDA** (Wei and Zou 2019) and **BT**(Back Translation) (Shleifer 2019); contextual augmentation using pre-trained laguage models by word modification, including **CBERT** (Wu et al. 2019a), **BERTexpand**, **BERTprepend** (Kumar, Choudhary, and Cho 2020), and by generation, including **GPT2**; representational data augmentation including **Mixup** (Zhang et al. 2018), **Textsmooth** (Wu et al. 2022), **ADV**(adversarial data augmentation) (Miyato, Dai, and Goodfellow 2017).

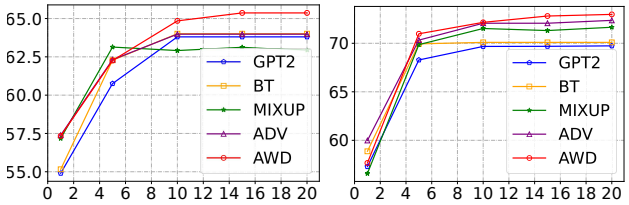
Implementation We implement our AWD model using Pytorch deep learning framework. The BERT-uncased Base model is used as the text classifier. The dimension of the word embedding d is set to 768. The dilution network for each class is implemented using an MLP followed by a sigmoid activation function. We train our model using Adam optimizer with default configurations. The learning rate is set to 5×10^{-4}. We train AWD and each baseline model for 30 epochs. We repeat all experiments 15 times and report their mean accuracy. We tune the hyper-parameters by grid search on the validation set. The hyper-parameter for training AWD(strict) is $\lambda = 1$ and $\rho = 0.3, 0.5, 0.3$ for respect k=10, 20, 50. When training AWD(strict), we perform 5 SGD updates in a dilution-network optimization step with a learning rate of 0.01. The hyper-parameter for training AWD(loose) is $\gamma = 0.005$. All experiments are conducted on an NVIDIA Tesla P100 GPU with 16GB memory.

Low-Resource Text Classification Results

Main Results The low-resource text classification results of AWD and baseline models are shown in Table 2. The results show that most contextual data augmentation methods and representational data augmentation methods improve upon the BERT baselines. The representational data augmentations are relatively better than the contextual data augmentation methods. These observations manifest the effectiveness of representational data augmentations. Compared to existing representational data augmentation methods Mixup, Textsmooth and ADV, our AWD(strict) and AWD(loose) achieve better results in all settings on SST-2, TREC, Average and in the $k = 50$ setting on SNIPS. These results demonstrate that our methods AWD(strict) and AWD(loose) significantly improve generalization in low-resource text classification and build themselves as the new state-of-the-art methods of text data augmentation.

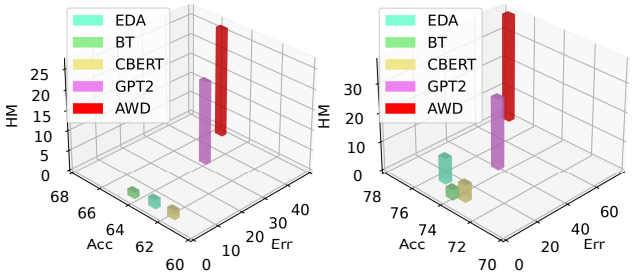
Method	SST-2			TREC			SNIPS			Average		
	k=10	k=20	k=50	k=10	k=20	k=50	k=10	k=20	k=50	$k=10$	$k=20$	$k=50$
BERT	62.2(7.1)	71.9(4.8)	79.7(4.3)	72.1(10.5)	82.9(4.9)	88.4(3.4)	90.6(1.3)	92.9(1.4)	94.7(0.9)	74.9(6.3)	82.6(3.7)	87.6(2.9)
EDA	62.7(8.5)	71.5(6.1)	80.3(3.1)	75.0(7.5)	80.8(4.6)	86.6(3.9)	90.2(2.0)	93.1(1.2)	94.2(1.3)	75.9(6.0)	81.8(4.0)	87.1(2.8)
BT	64.0(7.8)	70.1(6.8)	80.8(3.8)	73.7(7.8)	82.1(5.9)	88.5(2.3)	91.0(1.5)	93.2(0.9)	94.6(0.8)	76.3(5.7)	81.8(4.6)	88.0(2.3)
CBERT	61.4(7.9)	72.6(6.6)	80.2(3.1)	73.2(7.9)	82.7(5.0)	88.7(2.7)	90.1(2.3)	93.2(1.0)	94.3(1.2)	74.9(6.1)	82.9(4.2)	87.7(2.3)
BERTexpand	62.4(8.3)	71.1(5.2)	80.3(4.5)	75.3(6.4)	83.3(4.6)	86.7(4.0)	90.5(2.1)	93.1(1.3)	94.8(1.0)	76.0(5.6)	82.5(3.7)	87.3(3.2)
BERTprepend	63.8(7.6)	70.9(5.4)	78.8(6.2)	73.1(7.5)	81.4(4.4)	86.0(3.7)	90.4(1.7)	93.4(1.1)	94.9(1.1)	75.7(5.6)	81.9(3.6)	86.6(3.6)
GPT2	63.8(5.4)	69.7(5.9)	75.7(5.0)	74.2(7.2)	80.2(6.9)	84.8(4.1)	90.3(1.4)	93.2(0.6)	93.5(1.3)	76.1(4.7)	81.0(4.5)	84.7(3.5)
Mixup	64.4(5.8)	72.6(4.0)	80.3(6.8)	73.6(5.7)	81.3(6.0)	87.8(4.2)	**91.4**(1.1)	**93.9**(1.2)	94.8(0.8)	76.5(4.2)	82.6(3.7)	87.6(3.9)
Textsmooth	63.1(7.6)	72.5(5.6)	82.5(2.8)	75.3(6.7)	81.3(5.8)	88.1(3.9)	90.5(1.2)	93.7(1.0)	94.8(1.0)	76.3(5.2)	82.5(4.2)	88.5(2.6)
ADV	64.0(7.7)	72.4(8.1)	81.3(3.9)	74.9(9.6)	82.6(6.4)	88.0(4.7)	90.5(2.4)	93.5(0.9)	94.8(0.7)	76.5(6.6)	82.8(5.1)	88.0(3.1)
AWD(strict)	**65.4**(6.8)	72.9(5.8)	82.7(3.0)	76.7(7.3)	**83.7**(4.4)	89.1(3.4)	91.2(1.8)	93.8(1.0)	95.0(1.0)	77.7(5.3)	83.5(3.7)	88.9(2.5)
AWD(loose)	65.2(7.8)	**73.9**(5.4)	**82.7**(5.2)	**77.1**(7.1)	83.4(4.2)	**90.0**(3.1)	91.0(1.9)	93.7(1.2)	**95.3**(0.7)	**77.8**(5.6)	**83.7**(3.6)	**89.3**(3.0)

Table 2: The evaluation results on SST-2, TREC and SNIPS. The bold and underline indicate the best and second-best results.



(a) k=10 (b) k=20

Figure 4: Results during the training. The horizontal and vertical axes respectively denote the epochs and accuracy.



(a) SST-2 (b) TREC

Figure 5: The hardness analysis of different text data augmentation methods on the SST-2 and TREC datasets.

The Effectiveness of Different Data Augmentations To investigate the effectiveness of training models using different data augmentations, we compare the evaluation accuracy after different epochs during the training process on SST-2 in Figure 4. As shown in the figure, our AWD generally achieves the best accuracy as the training progresses in both $k = 10$ and $k = 20$ settings, which manifests the effectiveness of AWD data augmentations. Another observation is that most data augmentations except AWD converge near 10 epochs and their accuracy stops increasing. Our AWD continues to increase the model performance after 10 epochs. This observation demonstrates that AWD can produce more effective text data augmentations that boost the model training. It also implicitly suggests that AWD may produce hard positive examples that benefit the training of the later stage.

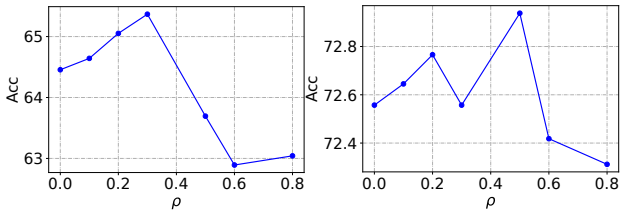
More Detailed Analysis on AWD

The Hardness of Different Data Augmentations We train a BERT model using the whole training set excluding the low-resource data as a discriminator to measure the hardness of given data augmentations, i.e., a higher error rate on this discriminator implies that the examples are harder. To appropriately evaluate all methods, we compute the harmonic mean (HM), the integrated performance of classification accuracy (Acc) and error rate on the discriminator (Err). Harder augmentations and better accuracy result in High HM. High Err but low Acc reflect that a model may produce undesired hard negative augmentations. We compare different text data augmentations with $k = 10$ on SST-2 and TREC in Figure 5, illustrating that harder data augmentations may increase accuracy, but there are exceptions, e.g., GPT2 produces relatively harder data augmentations but has lower accuracy. It may be because it produces hard but negative examples that may harm the model training. Among the compared models, our AWD achieve the highest HM because of high Acc and Err, demonstrating that AWD effectively produces hard positive examples.

Extending AWD to New Examples To study the extensibility of AWD to new examples, we first pre-train the AWD(strict) and AWD(loose) models on $k = 10, 20, 50$ training sets, then fix their parameters and use their dilution networks to generate data augmentations for new examples in new training sets with respect $k = 10, 20, 50$, next train the BERT model on each new training sets together with the respectively generated data augmentations. Note that in this setup, the dilution weights are not learned from the training data but extended from the data used for AWD pre-training. The evaluation results of BERT trained with original data and extended augmented data are shown in Table 3, which presents that the data augmentations for new examples generated by our AWD models effectively improve the original BERT model. This suggests that AWD can extend well to new examples without further training.

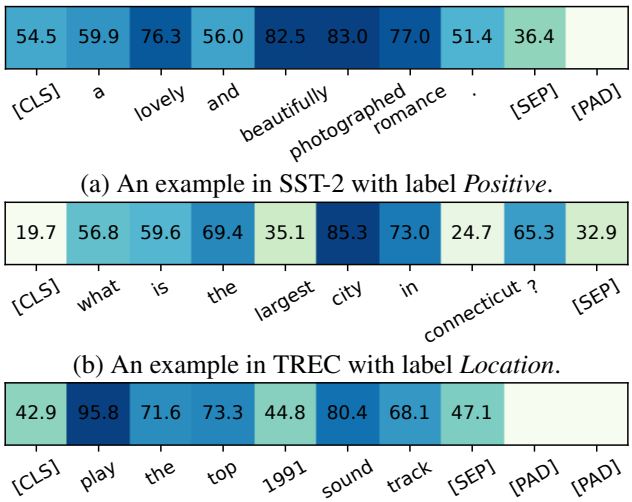
Method	SST-2			TREC			SNIPS			Average		
	k=10	k=20	k=50	k=10	k=20	k=50	k=10	k=20	k=50	$k=10$	$k=20$	$k=50$
BERT	62.2(7.1)	71.9(4.8)	79.7(4.3)	72.1(10.5)	82.9(4.9)	88.4(3.4)	90.6(1.3)	92.9(1.4)	94.7(0.9)	74.9(6.3)	82.6(3.7)	87.6(2.9)
BERT+strict	**63.2**(7.8)	72.7(6.2)	81.0(3.6)	**75.2**(9.3)	**83.5**(4.5)	**89.2**(3.2)	**90.9**(2.3)	93.3(1.1)	94.6(0.9)	**76.4**(6.5)	83.2(3.9)	88.3(2.6)
BERT+loose	63.2(7.5)	**74.2**(5.7)	**82.5**(2.2)	74.8(9.7)	83.2(4.5)	88.9(2.2)	90.8(1.7)	**93.4**(1.0)	**94.9**(1.1)	76.2(6.3)	**83.6**(3.7)	**88.8**(1.8)

Table 3: The results of applying AWD models to new examples. BERT+strict and BERT+loose denote the BERT model trained with data augmentations for new examples generated by pre-trained AWD(strict) and AWD(loose), respectively.



(a) k=10 (b) k=20

Figure 6: The hyper-parameter analysis of ρ on SST-2.



(a) An example in SST-2 with label *Positive*.

(b) An example in TREC with label *Location*.

(c) An example in SNIPS with label *PlayMusic*.

Figure 7: The selected examples from the three datasets, the values indicate the corresponding dilution weights. We pad the three sentences to the same length for a better view.

The Hyper-Parameter Analysis of AWD As mentioned in the challenges, choosing appropriate dilution weights is essential to guarantee the effect of word dilution. Extreme dilution weights may lead the augmented texts to be useless or meaningless. To investigate this phenomenon, we illustrate the performance changes with various scopes of dilution weights by controlling the hyper-parameter ρ in the AWD(strict) model. We report the results on SST-2 using different hyper-parameter ρ in Figure 6. As shown in the figure, In both k=10 and k=20 settings, the model performance generally increases first and then decreases as the hyper-parameter ρ increase. The illustrated tendencies suggest our guess. When ρ is too small, the model generates low dilution weights and constructs data augmentations with minor-difference from the original texts that may bring slight improvement. When ρ is too large, the model generates high dilution weights, making the augmented texts become meaningless and may degrade the performance.

The Interpretability of AWD To demonstrate the interpretability of AWD, we make case studies on the dilution weights assigned to each word of some selected examples. We select three text examples, each from one of the SST-2, TREC, SNIPS datasets, and visualize their learned dilution weights in Figure 7. In the (a) case from the SST-2 dataset, the words *lovely*, *beautifully*, *photographed* and *romance* all express positive sentiment and they are semantically related to the label *Positive* of the text. These words are respectively assigned with high dilution weights 0.763, 0.825, 0.830 and 0.770. This case manifests that our AWD model effectively learns the interpretable dilution weights through adversarial optimization. In the (b) case from the TREC dataset, the word *city* related to the label *Location* is assigned with a high dilution weight 0.853. However, the word *connecticut* also related to *Location* is assigned a low dilution weight 0.247. This may be because this word is a low-frequency knowledge word whose semantics are difficult to capture. In the (c) case from the SNIPS dataset, the words *play* and *sound* related to the label *PlayMusic* are assigned with high dilution weights. All special words, such as *[CLS]*, *[SEP]* are assigned with relatively low dilution weights. These cases demonstrate that the dilution weights generated by our AWD model are semantically interpretable.

Conclusion

This work proposes a new text data augmentation method, Adversarial Word Dilution (AWD), for low-resource text classification. It can generate hard positive examples as data augmentations while keeping interpretability and extensibility. AWD constructs data augmentations through word dilution by weighted mixing the embedding of strong positive words with unknown-word embedding. We adversarially train AWD with constrained min-max optimization by iteratively learning the dilution weights and training the classifier. Experiment results show that AWD outperforms the state-of-the-art text data augmentation methods and demonstrates its interpretability and extensibility. We hope AWD will extend to a broad spectrum of other text classification tasks with limited labeled data, such as few-shot text classification or semi-supervised text classification.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0110700, in part by the Fundamental Research Funds for the Central Universities, in part by the State Key Laboratory of Software Development Environment.

References

Anaby-Tavor, A.; Carmeli, B.; Goldbraich, E.; Kantor, A.; Kour, G.; Shlomov, S.; Tepper, N.; and Zwerdling, N. 2020. Do Not Have Enough Data? Deep Learning to the Rescue! In *AAAI*, 7383–7390.

Andreas, J. 2020. Good-Enough Compositional Data Augmentation. In *ACL*, 7556–7566.

Artetxe, M.; Labaka, G.; Agirre, E.; and Cho, K. 2018. Unsupervised Neural Machine Translation. In *ICLR*.

Chen, J.; Shen, D.; Chen, W.; and Yang, D. 2021. HiddenCut: Simple Data Augmentation for Natural Language Understanding with Better Generalization. *CoRR*.

Chen, J.; Yang, Z.; and Yang, D. 2020. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *ACL*, 2147–2157.

Chen, J.; Zhang, R.; Mao, Y.; and Xu, J. 2022. ContrastNet: A Contrastive Learning Framework for Few-Shot Text Classification. In *AAAI*, 10492–10500.

Chen, L.; Ruan, W.; Liu, X.; and Lu, J. 2020. SeqVAT: Virtual Adversarial Training for Semi-Supervised Sequence Labeling. In *ACL*, 8801–8811.

Chen, M.; Tang, Q.; Wiseman, S.; and Gimpel, K. 2019. Controllable Paraphrase Generation with a Syntactic Exemplar. In *ACL*, 5972–5984.

Cheng, Y.; Jiang, L.; and Macherey, W. 2019. Robust Neural Machine Translation with Doubly Adversarial Inputs. In *ACL*, 4324–4333.

Cheng, Y.; Jiang, L.; Macherey, W.; and Eisenstein, J. 2020. AdvAug: Robust Adversarial Augmentation for Neural Machine Translation. In *ACL*, 5961–5970.

Coucke, A.; Saade, A.; Ball, A.; Bluche, T.; Caulier, A.; Leroy, D.; Doumouro, C.; Gisselbrecht, T.; Caltagirone, F.; Lavril, T.; Primet, M.; and Dureau, J. 2018. Snips Voice Platform: an embedded Spoken Language Understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 4171–4186.

Dopierre, T.; Gravier, C.; and Logerais, W. 2021. ProtAugment: Unsupervised diverse short-texts paraphrasing for intent detection meta-learning. In *ACL*, 2454–2466.

Edunov, S.; Ott, M.; Auli, M.; and Grangier, D. 2018. Understanding Back-Translation at Scale. In *EMNLP*, 489–500.

Fadaee, M.; Bisazza, A.; and Monz, C. 2017. Data Augmentation for Low-Resource Neural Machine Translation. In *ACL*, 567–573.

Gao, F.; Zhu, J.; Wu, L.; Xia, Y.; Qin, T.; Cheng, X.; Zhou, W.; and Liu, T. 2019. Soft Contextual Data Augmentation for Neural Machine Translation. In *ACL*, 5539–5544.

Garg, S.; and Ramakrishnan, G. 2020. BAE: BERT-based Adversarial Examples for Text Classification. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *EMNLP*, 6174–6181.

Geng, R.; Li, B.; Li, Y.; Zhu, X.; Jian, P.; and Sun, J. 2019. Induction Networks for Few-Shot Text Classification. In *EMNLP-IJCNLP*, 3902–3911.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative Adversarial Nets. In *NIPS*, 2672–2680.

Guo, D.; Kim, Y.; and Rush, A. M. 2020. Sequence-Level Mixed Sample Data Augmentation. In *EMNLP*, 5547–5552.

Hsu, W.; Tang, H.; and Glass, J. R. 2018. Unsupervised Adaptation with Interpretable Disentangled Representations for Distant Conversational Speech Recognition. In *INTERSPEECH*, 1576–1580.

Hu, Z.; Tan, B.; Salakhutdinov, R.; Mitchell, T. M.; and Xing, E. P. 2019. Learning Data Manipulation for Augmentation and Weighting. In *NeurIPS*, 15738–15749.

Iyyer, M.; Manjunatha, V.; Boyd-Graber, J. L.; and III, H. D. 2015. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *ACL*, 1681–1691.

Jia, R.; and Liang, P. 2016. Data Recombination for Neural Semantic Parsing. In *ACL*.

Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Zhao, T. 2020. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization. In *ACL*, 2177–2190.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised Contrastive Learning. In *NeurIPS*.

Kobayashi, S. 2018. Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations. In *NAACL-HLT*, 452–457.

Kolomiyets, O.; Bethard, S.; and Moens, M. 2011. Model-Portability Experiments for Textual Temporal Analysis. In *ACL*, 271–276.

Kumar, A.; Bhattamishra, S.; Bhandari, M.; and Talukdar, P. P. 2019. Submodular Optimization-based Diverse Paraphrasing and its Effectiveness in Data Augmentation. In *NAACL-HLT*, 3609–3619.

Kumar, V.; Choudhary, A.; and Cho, E. 2020. Data Augmentation using Pre-trained Transformer Models. *CoRR*, abs/2003.02245.

Lee, J. H.; Ko, S.; and Han, Y. 2021. SALNet: Semi-supervised Few-Shot Text Classification with Attention-based Lexicon Construction. In *AAAI*, 13189–13197.

Li, X.; and Roth, D. 2002. Learning Question Classifiers. In *COLING*.

Miao, Z.; Li, Y.; Wang, X.; and Tan, W. 2020. Snippext: Semi-supervised Opinion Mining with Augmented Data. In *WWW*, 617–628.

Miyato, T.; Dai, A. M.; and Goodfellow, I. J. 2017. Adversarial Training Methods for Semi-Supervised Text Classification. In *ICLR*.

Pang, B.; and Lee, L. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL*, 115–124.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *ACL*, 1085–1097.

Schroff, F.; Kalenichenko, D.; and Philbin, J. 2015. FaceNet: A unified embedding for face recognition and clustering. In *CVPR*, 815–823.

Shleifer, S. 2019. Low Resource Text Classification with ULMFit and Backtranslation. *CoRR*, abs/1903.09244.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *ACL*, 1631–1642.

Wang, W. Y.; and Yang, D. 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 *EMNLP*, 2557–2563.

Wei, J. W.; and Zou, K. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *EMNLP-IJCNLP*, 6381–6387.

Wu, X.; Gao, C.; Lin, M.; Zang, L.; and Hu, S. 2022. Text Smoothing: Enhance Various Data Augmentation Methods on Text Classification Tasks. In *ACL*, 871–875.

Wu, X.; Lv, S.; Zang, L.; Han, J.; and Hu, S. 2019a. Conditional BERT Contextual Augmentation. In *ICCS*, 84–95.

Wu, Z.; Wang, S.; Qian, Y.; and Yu, K. 2019b. Data Augmentation Using Variational Autoencoder for Embedding Based Speaker Verification. In *INTERSPEECH*, 1163–1167.

Xie, Q.; Dai, Z.; Hovy, E. H.; Luong, T.; and Le, Q. 2020. Unsupervised Data Augmentation for Consistency Training. In *NeurIPS*.

Xie, Z.; Wang, S. I.; Li, J.; Lévy, D.; Nie, A.; Jurafsky, D.; and Ng, A. Y. 2017. Data Noising as Smoothing in Neural Network Language Models. In *ICLR*.

Yang, Y.; Malaviya, C.; Fernandez, J.; Swayamdipta, S.; Bras, R. L.; Wang, J.; Bhagavatula, C.; Choi, Y.; and Downey, D. 2020. G-DAug: Generative Data Augmentation for Commonsense Reasoning. In *EMNLP*, volume EMNLP 2020 of *Findings of ACL*, 1008–1025.

Yu, M.; Guo, X.; Yi, J.; Chang, S.; Potdar, S.; Cheng, Y.; Tesauro, G.; Wang, H.; and Zhou, B. 2018. Diverse Few-Shot Text Classification with Multiple Metrics. In *NAACL-HLT*, 1206–1215.

Zhang, H.; Cissé, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

Zhang, R.; Yu, Y.; and Zhang, C. 2020. SeqMix: Augmenting Active Sequence Labeling via Sequence Mixup. In *EMNLP*, 8566–8579.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. In *NIPS*, 649–657.

Zhu, C.; Cheng, Y.; Gan, Z.; Sun, S.; Goldstein, T.; and Liu, J. 2020. FreeLB: Enhanced Adversarial Training for Natural Language Understanding. In *ICLR*.