EPiDA: An Easy Plug-in Data Augmentation Framework for High Performance Text Classification

Supplementary File

Minyi Zhao[1] **Lu Zhang**[1] **Yi Xu**[1]
Jiandong Ding[2] **Jihong Guan**[3] **Shuigeng Zhou**[1*]
[1]Shanghai Key Lab of Intelligent Information Processing, and School of Computer Science, Fudan University, China
[2]Alibaba Group [3]Tongji University, China
[1,2]{zhaomy20, l_zhang19, yxu17, jdding, sgzhou}@fudan.edu.cn [3]jhguan@tongji.edu.cn

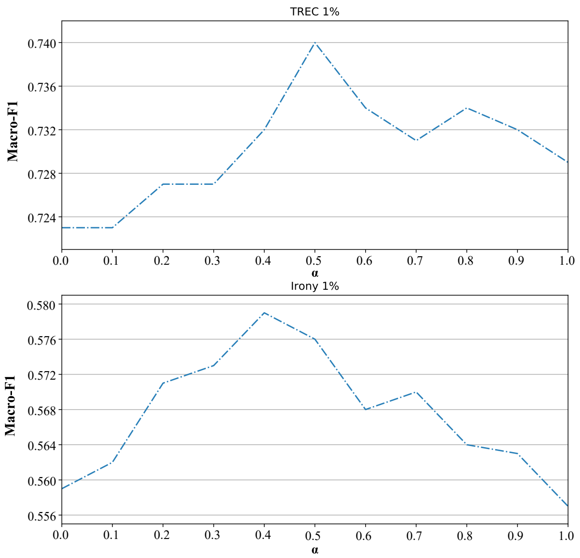

Figure 1: The Macro-F1 classification performances under different α. Top: **TREC** 1%, Bottom: **Irony** 1%.

1 Weighted Addition of REM and CEM

Here we discuss the usage of weighted addition to combine REM and CEM. That is to say we introduce an additional hyperparameter α, $\alpha \in [0, 1]$ to control the trade-off of REM and CEM:

$$s_{tot} = \alpha s_{div} + (1 - \alpha) s_{qua} \tag{1}$$

A larger α highlights diversity and suppresses quality, and vice versa.

We discuss the influence of α on two datasets: **TREC** and **Irony**. We also use CNN (Kim, 2014) as the classifier and Macro-F1 as the metric and report the average results over five times repeated experiments. The classification performance under different α is presented in Fig. 1.

As shown in Fig. 1, in **TREC** and **Irony** tasks, the best values of α are 0.5 and 0.4, respectively. Although $\alpha = 0.4$ (0.579 vs. 0.576) performs better on the **Irony** task, 0.5 is sufficient to achieve satisfactory results on both tasks. Ergo, we set α to 0.5 in this paper.

*Corresponding author.

Loss Function	TREC 1%	Irony 1%
Eq. (2)	0.722	0.534
Eq. (3)	0.736	0.474
Eq. (4)	0.740	0.576

Table 1: Ablation study of different loss functions at TERC 1% and Irony 1%. The results are reported by Macro-F1 under five times repeated experiments.

2 Ablation Study on Loss Function

Here we take an ablation study to support the combined loss function used in our paper. Actually, they are three loss functions in this paper.

The first one is the original loss function without performing DA

$$L_o(\omega) = \frac{1}{n} \sum_{i=1}^{n} l(\omega^\top \phi(x_i); y_i), \tag{2}$$

which means we do not take DA to enrich the training data.

The second is the new loss function after using DA:

$$L_g(\omega) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{m} \sum_{j=1}^{m} l(\omega^\top \phi(t_i^j); y_i). \tag{3}$$

The third is the combined loss function:

$$L(\omega) = L_o(\omega) + L_g(\omega). \tag{4}$$

The experimental results of different loss functions at **TERC** 1% and **Irony** 1% is presented in Tab. 1. The combined loss function Eq. (4) outperforms Eq. (2) and Eq. (3) in **TERC** 1% and **Irony** 1%. As mentioned earlier (*c.f.* Visualization study in main paper), the samples augmented by EPiDA are more diverse than the original samples, which

EDA	+EPiDA	CWE	+ EiPDA	DataBoost
188.4	43.8	30.7	10.1	1.0

Table 2: Generation speed comparison with existing DA methods. The speed is measured by the samples generated per Second. Except for DataBoost whose data are cited from (Liu et al., 2020), all the other methods' results are obtained on a NVIDIA RTX 3090.

K	2	3	5	7	10
Macro-F1	0.573	0.577	0.576	0.574	0.575

Table 3: Comparison of the classification performance on Irony 1% under different amplification factor K values.

also causes a deviation. Such deviation limits the classification performance. However, the combined loss function Eq. (4) solved this problem by mixing the augmented samples and the original samples.

3 Generation Speed

Tab. 2 presents the results of generation speed of EPiDA. We evaluate the speed by the number of samples generated by a DA algorithm per second. As shown in Tab. 2, after using EPiDA, EDA and CWE are still faster than DataBoost.

4 Effect of the amplification factor K.

By grid search, we present the performance results of different K values in Tab. 3, from which we set K to 3 in our experiments.

5 More Verification of EPiDA

In order to fully demonstrate the performance of EPiDA, we additionally follow the experimental settings of (Shi et al., 2021) and compare our method with SUB^2. The dataset and classifier in this experiment is SST (Socher et al., 2013) and XLM-R (Conneau et al., 2019), respectively. Following (Shi et al., 2021), to avoid over-fitting to the small development set and tuning on test set issues, we introduce small "development test" (devtest) sets for SST, and only evaluate on the test sets using classifiers with the best devtest performance. The experimental results are placed in Tab. 4. As shown in Tab. 4, after introducing EPiDA, the performance of EDA and CWE are improved. Besides, our method can also achieve comparable performance with SUB^2 in SST task, which demonstrates the superiority of our framework.

Method	Accuracy
SST-10% $(\lvert\mathcal{D}_{train}\rvert = 0.8\text{K}, \lvert\mathcal{D}_{devtest}\rvert = 0.1\text{K})$	
NOAUG	25.4
EDA (Wei and Zou, 2019)	40.6
CWE (Kobayashi, 2018)	44.9
SUB^2 (Shi et al., 2021)	45.8
EPiDA+EDA	43.5
EPiDA+CWE	**45.9**

Table 4: Accuracy ($\times$100) on the SST standard test set. The best numbers in each section are bolded.

Method	Corpus	
	AGNews	MR
BERT	0.944	0.868
VDA	0.945	0.878
EPiDA with EDA	**0.949**	**0.879**

Table 5: Accuracy ($\times$100) on the test sets of AGNews and MR. The best numbers in each section are bolded.

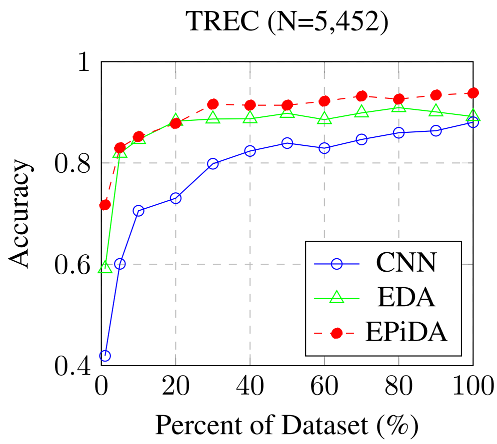

Figure 2: Classification accuracy w/o EPiDA for various original data sizes (before DA) used for training.

We also provide the experimental results following the setting of VDA (Zhou et al., 2021) in AGNews (Zhang et al., 2015) and MR (Pang and Lee, 2005) corpus. We take BERT as classifier, the experimental results are placed in Tab. 5. As shown in Tab. 5, EPiDA outperforms VDA in classification accuracy.

6 Apply EPiDA performs in high-resource settings

In Fig. 2 we provide classification performance vs. original training data size. EPiDA performs well in low-resource settings. However, ever when all the data are used, EPiDA still boosts accuracy (CNN: 0.88, EDA: 0.89, ours: 0.93).

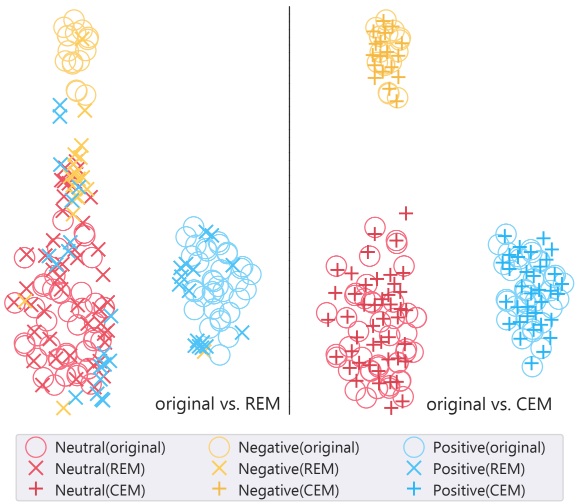

Figure 3: t-SNE hidden state visualization results of Sentiment Analysis task. Left: original vs. REM only; Right: original vs. CEM only. Different colors represent different classes (Neutral, Negative, and Positive), while different shapes represent different augmentation algorithms (original, REM only and CEM only).

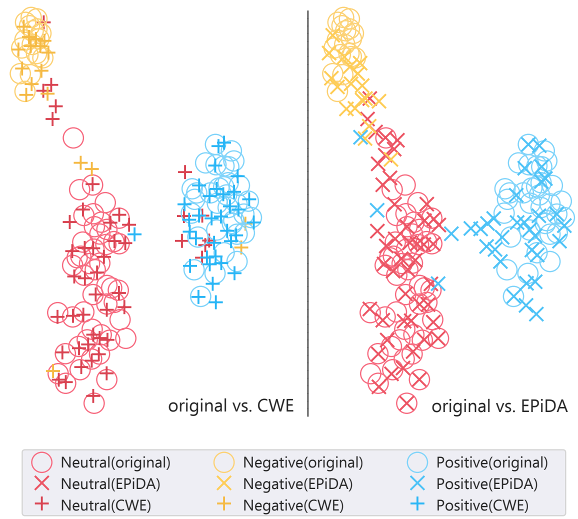

Figure 4: t-SNE hidden state visualization results of Sentiment Analysis task. Left: original vs. CWE; Right: original vs. EPiDA+CWE. Different colors represent different classes (Neutral, Negative, and Positive), while different shapes represent different augmentation algorithms (original, CWE only and EPiDA+CWE).

7 Visualization Case of REM and CEM

Here we provide the visualization results of using REM and CEM separately to illustrate the benefits of REM and CEM more clearly. As shown in Fig. 3, REM encourages to enhance diversity while low-quality samples with wrong labels will be generated. In contrast, CEM encourages to generate high-quality while less-diversity samples.

8 Visualization Case of CWE

Fig. 4 show the visualization result of CWE (Kobayashi, 2018) and EiPDA+CWE. Similar conclusions can also be drawn from Fig. 4. CWE itself has the ability of enhancing diversity, and with the help of EPiDA, the quality of the generation has been dramatically improved (See Positive Class).

9 Core Implementation Code

The implementation of REM and CEM is available at Fig. 5. Here, the calculation of mutual information refers to (Ji et al., 2019).

10 Replacement of REM and CEM

Here we discuss the replacement of REM and CEM. In other words, we separately use PPL or cosine similarity mentioned in (Zuo et al., 2021) to replace REM or CEM to control diversity or quality. The experimental results are presented in Tab. 6. As shown in Tab. 6, REM+CEM outperforms other variants, which demonstrates the superiority of our method.

11 More Implementation Details

Here we supply additional details of our implementation.

Dataset Preprocessing: We clean all punctuation, stop words, hashtags, numbers and URL links in the tweets corpora.

Data Augmentation Algorithms: There are three DA algorithms used in this paper: EDA (Wei and Zou, 2019)[1], CWE (Kobayashi, 2018)[2], and TextAttack (Morris et al., 2020)[3].

Classifiers: Here we provide the implementation of the classifiers. There are four classifiers used in our paper: CNN (Kim, 2014)[4], BERT (Devlin et al., 2019)[5], XLNet (Yang et al., 2019)[6] and XLM-R (Conneau et al., 2019)[7].

Random Seeds: The random seeds used in this paper are 0,1,2,3 and 4, respectively.

[1]https://github.com/jasonwei20/eda_nlp
[2]https://github.com/makcedward/nlpaug
[3]https://github.com/QData/TextAttack
[4]https://github.com/galsang/CNN-sentence-classification-pytorch
[5]https://huggingface.co/transformers/model_doc/bert.html
[6]https://huggingface.co/transformers/model_doc/xlnet.html
[7]https://huggingface.co/transformers/model_doc/xlmroberta.html

REM	PPL	CEM	CosSim	TREC 1%	Irony 1%
✓	-	✓	-	**0.740**	**0.576**
-	✓	✓	-	0.731	0.567
✓	-	-	✓	0.736	0.566
-	✓	-	✓	0.730	0.562

Table 6: Ablation study of the replacement of REM and CEM at TERC 1% and Irony 1%. The results are reported by Macro-F1 under five times repeated experiments.

```
EPS = 1e-10
def REM(z, zt):
z[(z < EPS).data] = EPS
return -torch.sum(zt*torch.log(z))
def MI(z, zt):
C = zt.size()[1]
P = (z.unsqueeze(2) * zt.unsqueeze(1)).sum(dim=0)
P = ((P + P.t()) / 2) / P.sum()
P[(P < EPS).data] = EPS
Pi = P.sum(dim=1).view(C, 1).expand(C, C)
Pj = P.sum(dim=0).view(1, C).expand(C, C)
return 1.0 - (P * (log(Pi) + log(Pj) -
 log(P))).sum()
def H(z):
z[(z < EPS).data] = EPS
return -(z*torch.log(z)).sum()
def CEM(z, zt):
return MI(z, zt) - H(z)
```

Figure 5: Python implementation of REM and CEM. z is the probability distribution predicted by the classifier, and z_t is the probability distribution of original sample.

Others: We take AdamW (Loshchilov and Hutter, 2018) as the optimizer. All the experiments are conducted at 4 NVIDIA RTX 3090 GPUs with Pytorch1.8.

12 Limitation

The major limitation of EPiDA is the training time. Although EPDA can bring performance improvements, it will reduce the training speed by at least K(the amplification factor) times. This means that when the DA method and the classifier itself are cumbersome, the overall training time will be long. Besides, how to measure or define samples' value is still an open problem.

13 Supplementary Example

In Tab. 7, we provide several detailed augmentation results of EPiDA. m and K are set to 3. Therefore, 9 candidate samples will be generated.

References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Xu Ji, Joao F Henriques, and Andrea Vedaldi. 2019. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9865–9874.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

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

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020. Data boost: Text data augmentation through reinforcement learning guided conditional generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9031–9041.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *Proceedings of the 2020 EMNLP*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124.

Task/Selected	Sentence	s_{div}	s_{qua}	s_{tot}
Sentiment	I'm about to eat four hot dogs and watch Miss USA. Happy Sunday.	0.00	1.00	1.00
	I am about to eat four hot dogs and watch Miss usa. Happy Sunday.	0.10	0.83	0.93
	I'm about to eat four track hot dogs and watch Miss USA. Happy Sunday.	0.17	0.70	0.87
✓	I'm about to eat four hot dogs and watch Miss USA. Happy Sunday.	0.00	1.00	1.00
✓	I'm about to eat four hot live dogs and watch Miss USA. Happy Sunday.	0.06	0.98	1.04
	I'm about to eat four hot dogs and watch Miss USA. Gold Sunday.	0.24	0.58	0.82
	I'm about to eat four hot dogs and watch Sun Miss USA. Happy Sunday.	0.00	0.98	0.98
✓	I'm about to eat hot dogs and watch Miss USA. Happy Sunday.	0.03	0.99	1.02
	I'm about to eat quadruplet hot dogs and watch Miss USA. Sunday.	1.00	0.00	1.00
	I'm about to eat four hot dogs and watch Miss USA. Happy Sunday.	0.00	1.00	1.00
Irony	A wonderful day of starting work at 6am	0.00	1.00	1.00
	A day wonderful of starting work at 6am	0.43	0.48	0.91
	A 6am day of starting work at wonderful	0.21	0.73	0.94
	A wonderful of starting work at 6am	0.75	0.18	0.93
	A grand day of starting work at 6am	0.39	0.53	0.92
✓	Day wonderful a of starting work at 6am	0.92	0.07	0.99
	A wonderful day starting of work at 6am	0.56	0.35	0.91
✓	A wonderful day of starting at 6am	1.00	0.00	1.00
✓	A day of starting work at 6am	0.87	0.12	0.99
	A wonderful day at starting work of 6am	0.75	0.18	0.93

Table 7: Some examples selected by SEAS. Underlined words are salient words. The first column will be checked if this augmented sample is seleceted by SEAS.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2021. Substructure substitution: Structured data augmentation for nlp. *arXiv preprint arXiv:2101.00411*.

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*, pages 1631–1642.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

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

Kun Zhou, Wayne Xin Zhao, Sirui Wang, Fuzheng Zhang, Wei Wu, and Ji-Rong Wen. 2021. Virtual data augmentation: A robust and general framework for fine-tuning pre-trained models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3875–3887.

Xinyu Zuo, Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, Weihua Peng, and Yuguang Chen. 2021. Learnda: Learnable knowledge-guided data augmentation for event causality identification. *arXiv preprint arXiv:2106.01649*.