Appendix

1 Initialization of REDA

We initialized REDA with the following editing rate for SR, RS, RI, and RD respectively: 0.2, 0.2, 0.1, and 0.1. We applied Python rounding rule to calculate and perform the number of edits needed for each operation. That means, if the number of edits is less than or equal to 0.5, it will be rounded down to 0 and thus no editing operation will apply. To make our experiments more controlled and doable, (1) we made RM only randomly perform two of the other four editing operations with one edit each; (2) and every editing operation will produce up to 2 non-duplicated augmented texts, if the train set size is less than 50k; otherwise, there will only be one augmented text instead. Every augmented text was crossed paired with the other text that was the pair to the text being augmented with the original label kept for the augmented text pair. That means, the augmented text pairs double the number of augmented texts set for each text. These settings also apply for the ablation study.

The synonym dictionary for English comes from WordNet (Miller, 1995). The synonym dictionary for Chinese comes from multiple reputable sources through web scraping.

2 Model Training

Training Settings. We reused the three simple models already constructed using Baidu's deep learning framework paddle[1]. We trained all the models in Baidu Machine Learning CodeLab on its AI Studio[2] with Tesla V100 GPU and 32G RAM, which the author could use up to 70 hours per week.

Basic Architecture. All the models begin with an Embedding layer that outputs 128-dimensional word embeddings. Then, the word embeddings for the text pairs each go through an encoder so that the encoded embeddings for the text pairs have same output dimensions and can be concatenated along the last axis. The concatenated embeddings run through a Linear layer, a Tanh activation function, and another Linear layer that outputs two dimensional logits. The details of the encoder configurations used for the CBOW, CNN, and LSTM models can be found at the footnote.[3]

Hyperparameters. The models were trained with 64 mini batches, a fixed .0005 learning rate, and constantly 3 epochs. We used Adaptive Moment Estimation (Adam) as the optimizer and cross entropy as the loss function.

Notes. Unlike Wei and Zou (2019), (1) We did not utilize pretrained word embeddings for our models, which will make the effects of text perturbations complicated and less obvious. Plus, we believe for a DA approach to be generally effective, it should also work in a setting where resources for pretrained word embeddings are limited or unavailable. (2) We did not use EarlyStopping or other similar callbacks, because that might increase the experimental costs to a point that obstructs training. Also, the effect of such a callback should be trivial as most of our models overfitted with 3 epochs.

[1] https://github.com/PaddlePaddle/PaddleNLP/blob/develop/examples/text_matching/simnet

[2] https://aistudio.baidu.com/aistudio/index

[3] https://github.com/PaddlePaddle/PaddleNLP/blob/develop/paddlenlp/seq2vec/encoder.py

References

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

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.