EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks

Jason Wei[1,2] **Kai Zou**[3]
[1]Protago Labs Research, Tysons Corner, Virginia, USA
[2]Department of Computer Science, Dartmouth College
[3]Department of Mathematics and Statistics, Georgetown University
jason.20@dartmouth.edu kz56@georgetown.edu

arXiv:1901.11196v2 [cs.CL] 25 Aug 2019

Abstract

We present **EDA**: **e**asy **d**ata **a**ugmentation techniques for boosting performance on text classification tasks. EDA consists of four simple but powerful operations: synonym replacement, random insertion, random swap, and random deletion. On five text classification tasks, we show that EDA improves performance for both convolutional and recurrent neural networks. EDA demonstrates particularly strong results for smaller datasets; on average, across five datasets, training with EDA while using only 50% of the available training set achieved the same accuracy as normal training with all available data. We also performed extensive ablation studies and suggest parameters for practical use.

1 Introduction

Text classification is a fundamental task in natural language processing (NLP). Machine learning and deep learning have achieved high accuracy on tasks ranging from sentiment analysis (Tang et al., 2015) to topic classification (Tong and Koller, 2002), but high performance often depends on the size and quality of training data, which is often tedious to collect. Automatic data augmentation is commonly used in computer vision (Simard et al., 1998; Szegedy et al., 2014; Krizhevsky et al., 2017) and speech (Cui et al., 2015; Ko et al., 2015) and can help train more robust models, particularly when using smaller datasets. However, because it is challenging to come up with generalized rules for language transformation, universal data augmentation techniques in NLP have not been thoroughly explored.

Previous work has proposed some techniques for data augmentation in NLP. One popular study generated new data by translating sentences into French and back into English (Yu et al., 2018). Other work has used data noising as smoothing (Xie et al., 2017) and predictive language models for synonym replacement (Kobayashi, 2018). Although these techniques are valid, they are not often used in practice because they have a high cost of implementation relative to performance gain.

Operation	Sentence
None	A sad, superior human comedy played out on the back roads of life.
SR	A ***lamentable***, superior human comedy played out on the ***backward*** road of life.
RI	A sad, superior human comedy played out on ***funniness*** the back roads of life.
RS	A sad, superior human comedy played out on ***roads*** back ***the*** of life.
RD	A sad, superior human out on the roads of life.

Table 1: Sentences generated using EDA. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion.

In this paper, we present a simple set of universal data augmentation techniques for NLP called EDA (**e**asy **d**ata **a**ugmentation). To the best of our knowledge, we are the first to comprehensively explore text editing techniques for data augmentation. We systematically evaluate EDA on five benchmark classification tasks, showing that EDA provides substantial improvements on all five tasks and is particularly helpful for smaller datasets. Code is publicly available at `http://github.com/jasonwei20/eda_nlp`.

2 EDA

Frustrated by the measly performance of text classifiers trained on small datasets, we tested a number of augmentation operations loosely inspired by those used in computer vision and found that they helped train more robust models. Here, we present the full details of EDA. For a given sentence in the training set, we randomly choose and perform one of the following operations:

1. **Synonym Replacement (SR):** Randomly choose n words from the sentence that are not stop words. Replace each of these words with one of its synonyms chosen at random.
2. **Random Insertion (RI):** Find a random synonym of a random word in the sentence that is not a stop word. Insert that synonym into a random position in the sentence. Do this n times.
3. **Random Swap (RS):** Randomly choose two words in the sentence and swap their positions. Do this n times.
4. **Random Deletion (RD):** Randomly remove each word in the sentence with probability p.

Since long sentences have more words than short ones, they can absorb more noise while maintaining their original class label. To compensate, we vary the number of words changed, n, for SR, RI, and RS based on the sentence length l with the formula $n=\alpha l$, where α is a parameter that indicates the percent of the words in a sentence are changed (we use $p=\alpha$ for RD). Furthermore, for each original sentence, we generate n_{aug} augmented sentences. Examples of augmented sentences are shown in Table 1. We note that synonym replacement has been used previously (Kolomiyets et al., 2011; Zhang et al., 2015; Wang and Yang, 2015), but to our knowledge, random insertions, swaps, and deletions have not been extensively studied.

3 Experimental Setup

We choose five benchmark text classification tasks and two network architectures to evaluate EDA.

3.1 Benchmark Datasets

We conduct experiments on five benchmark text classification tasks: (1) **SST-2**: Stanford Sentiment Treebank (Socher et al., 2013), (2) **CR**: customer reviews (Hu and Liu, 2004; Liu et al., 2015), (3) **SUBJ**: subjectivity/objectivity dataset (Pang and Lee, 2004), (4) **TREC**: question type dataset (Li and Roth, 2002), and (5) **PC**: Pro-Con dataset (Ganapathibhotla and Liu, 2008). Summary statistics are shown in Table 5 in Supplemental Materials. Furthermore, we hypothesize that EDA is more helpful for smaller datasets, so we delegate the following sized datasets by selecting a random subset of the full training set with N_{train}={500, 2,000, 5,000, all available data}.

3.2 Text Classification Models

We run experiments for two popular models in text classification. **(1)** Recurrent neural networks (RNNs) are suitable for sequential data. We use a LSTM-RNN (Liu et al., 2016). **(2)** Convolutional neural networks (CNNs) have also achieved high performance for text classification. We implement them as described in (Kim, 2014). Details are in Section 9.1 in Supplementary Materials.

4 Results

In this section, we test EDA on five NLP tasks with CNNs and RNNs. For all experiments, we average results from five different random seeds.

4.1 EDA Makes Gains

We run both CNN and RNN models with and without EDA across all five datasets for varying training set sizes. Average performances (%) are shown in Table 2. Of note, average improvement was 0.8% for full datasets and 3.0% for N_{train}=500.

	Training Set Size			
Model	500	2,000	5,000	full set
RNN	75.3	83.7	86.1	87.4
+EDA	79.1	84.4	87.3	88.3
CNN	78.6	85.6	87.7	88.3
+EDA	80.7	86.4	88.3	88.8
Average	76.9	84.6	86.9	87.8
+EDA	79.9	85.4	87.8	**88.6**

Table 2: Average performances (%) across five text classification tasks for models with and without EDA on different training set sizes.

4.2 Training Set Sizing

Overfitting tends to be more severe when training on smaller datasets. By conducting experiments using a restricted fraction of the available training data, we show that EDA has more significant improvements for smaller training sets. We run both normal training and EDA training for the following training set fractions (%): {1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100}. Figure 1(a)-(e) shows performance with and without EDA for each dataset, and 1(f) shows the averaged performance across all datasets. The best average accuracy without augmentation, 88.3%, was achieved using 100% of the training data. Models trained using EDA surpassed this number by achieving an

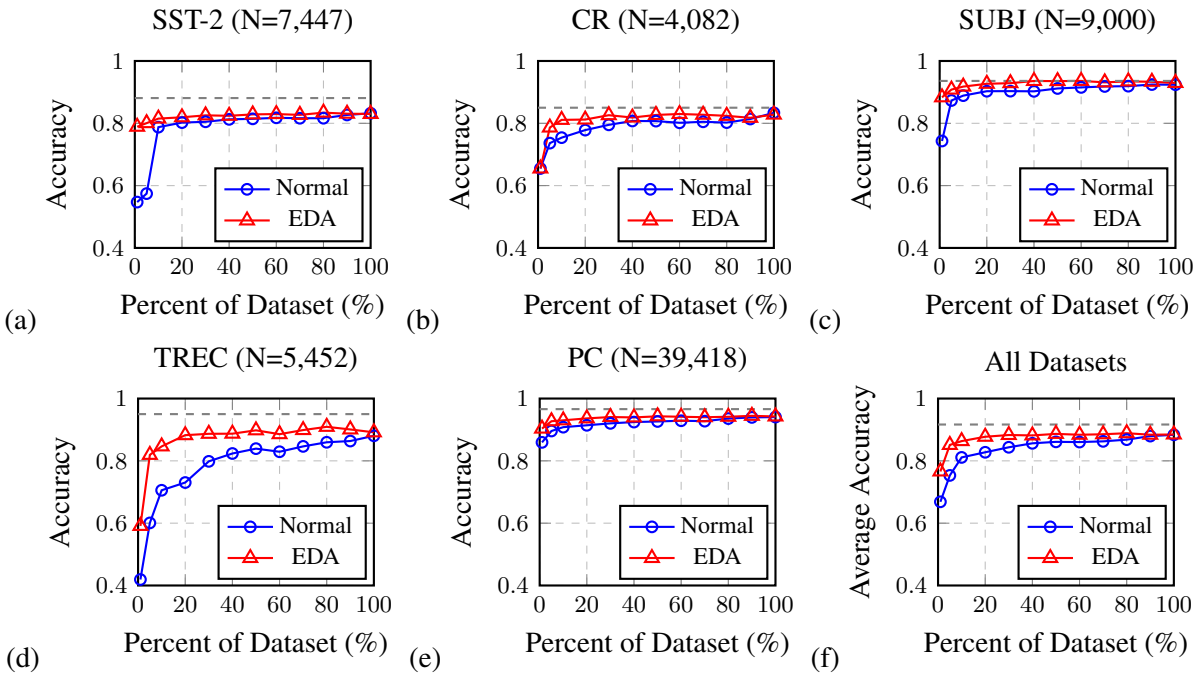

Figure 1: Performance on benchmark text classification tasks with and without EDA, for various dataset sizes used for training. For reference, the dotted grey line indicates best performances from Kim (2014) for SST-2, CR, SUBJ, and TREC, and Ganapathibhotla (2008) for PC.

average accuracy of 88.6% while only using 50% of the available training data.

4.3 Does EDA conserve true labels?

In data augmentation, input data is altered while class labels are maintained. If sentences are significantly changed, however, then original class labels may no longer be valid. We take a visualization approach to examine whether EDA operations significantly change the meanings of augmented sentences. First, we train an RNN on the pro-con classification task (PC) without augmentation. Then, we apply EDA to the test set by generating nine augmented sentences per original sentence. These are fed into the RNN along with the original sentences, and we extract the outputs from the last dense layer. We apply t-SNE (Van Der Maaten, 2014) to these vectors and plot their 2-D representations (Figure 2). We found that the resulting latent space representations for augmented sentences closely surrounded those of the original sentences, which suggests that for the most part, sentences augmented with EDA conserved the labels of their original sentences.

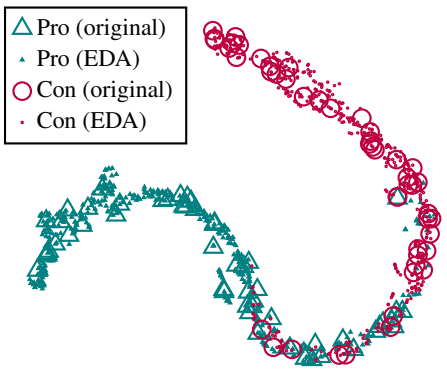

Figure 2: Latent space visualization of original and augmented sentences in the Pro-Con dataset. Augmented sentences (small triangles and circles) closely surround original sentences (big triangles and circles) of the same color, suggesting that augmented sentences maintianed their true class labels.

4.4 Ablation Study: EDA Decomposed

So far, we have seen encouraging empirical results. In this section, we perform an ablation study to explore the effects of each operation in EDA. Synonym replacement has been previously used (Kolomiyets et al., 2011; Zhang et al., 2015; Wang and Yang, 2015), but the other three EDA operations have not yet been explored. One could hypothesize that the bulk of EDA's performance gain is from synonym replacement, so we isolate each of the EDA operations to determine their individual ability to boost performance. For all four operations, we ran models using a single oper-

doit on déterminer le alpha par opération et la bonne combinaison par cross validation sur l'ensemble de train.

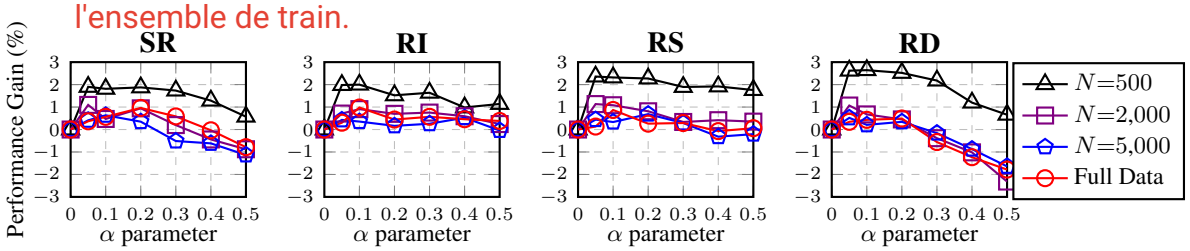

Figure 3: Average performance gain of EDA operations over five text classification tasks for different training set sizes. The α parameter roughly means "percent of words in sentence changed by each augmentation." SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion.

ation while varying the augmentation parameter α=$\{0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ (Figure 3).

It turns out that all four EDA operations contribute to performance gain. For SR, improvement was good for small α, but high α hurt performance, likely because replacing too many words in a sentence changed the identity of the sentence. For RI, performance gains were more stable for different α values, possibly because the original words in the sentence and their relative order were maintained in this operation. RS yielded high performance gains at $\alpha \leq 0.2$, but declined at $\alpha \geq 0.3$ since performing too many swaps is equivalent to shuffling the entire order of the sentence. RD had the highest gains for low α but severely hurt performance at high α, as sentences are likely unintelligible if up to half the words are removed. Improvements were more substantial on smaller datasets for all operations, and α=0.1 appeared to be a "sweet spot" across the board.

4.5 How much augmentation?

The natural next step is to determine how the number of generated augmented sentences per original sentence, n_{aug}, affects performance. In Figure 4, we show average performances over all datasets for n_{aug}=$\{1, 2, 4, 8, 16, 32\}$. For smaller training sets, overfitting was more likely, so generating many augmented sentences yielded large performance boosts. For larger training sets, adding more than four augmented sentences per original sentence was unhelpful since models tend to generalize properly when large quantities of real data are available. Based on these results, we recommend usage parameters in Table 3.

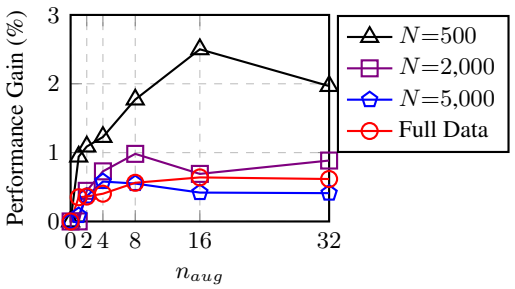

Figure 4: Average performance gain of EDA across five text classification tasks for various training set sizes. n_{aug} is the number of generated augmented sentences per original sentence.

N_{train}	α	n_{aug}
500	0.05	16
2,000	0.05	8
5,000	0.1	4
More	0.1	4

Table 3: Recommended usage parameters.

5 Comparison with Related Work

Related work is creative but often complex. Back-translation (Sennrich et al., 2016), translational data augmentation (Fadaee et al., 2017), and noising (Xie et al., 2017) have shown improvements in BLEU measure for machine translation. For other tasks, previous approaches include task-specific heuristics (Kafle et al., 2017) and back-translation (Silfverberg et al., 2017; Yu et al., 2018). Regarding synonym replacement (SR), one study showed a 1.4% F1-score boost for tweet classification by finding synonyms with k-nearest neighbors using word embeddings (Wang and Yang, 2015). Another study found no improvement in temporal analysis when replacing headwords with synonyms (Kolomiyets et al., 2011), and mixed results were reported for using SR in character-level text classification (Zhang et al., 2015); however, neither work conducted extensive ablation studies.

Most studies explore data augmentation as a complementary result for translation or in a task-specific context, so it is hard to directly compare

EDA with previous literature. But there are two studies similar to ours that evaluate augmentation techniques on multiple datasets. Hu (2017) proposed a generative model that combines a variational auto-encoder (VAE) and attribute discriminator to generate fake data, demonstrating a 3% gain in accuracy on two datasets. Kobayashi (2018) showed that replacing words with other words that were predicted from the sentence context using a bi-directional language model yielded a 0.5% gain on five datasets. However, training a variational auto-encoder or bidirectional LSTM language model is a lot of work. EDA yields results on the same order of magnitude but is much easier to use because it does not require training a language model and does not use external datasets. In Table 4, we show EDA's ease of use compared with other techniques.

Technique (#datasets)	LM	Ex Dat
Trans. data aug.[1] (1)	yes	yes
Back-translation[2] (1)	yes	yes
VAE + discrim.[3] (2)	yes	yes
Noising[4] (1)	yes	no
Back-translation[5] (2)	yes	no
LM + SR[6] (2)	yes	no
Contextual aug.[7] (5)	yes	no
SR - kNN[8] (1)	no	no
EDA (5)	**no**	**no**

Table 4: Related work in data augmentation. #datasets: number of datasets used for evaluation. Gain: reported performance gain on all evaluation datasets. LM: requires training a language model or deep learning. Ex Dat: requires an external dataset.[9]

6 Discussion and Limitations

Our paper aimed to address the lack of standardized data augmentation in NLP (compared to vision) by introducing a set of simple operations that might serve as a baseline for future investigation. With the rate that NLP research has progressed in recent years, we suspect that researchers will soon find higher-performing augmentation techniques that will also be easy to use.

Notably, much of the recent work in NLP focuses on making neural models larger or more complex. Our work, however, takes the opposite approach. We introduce simple operations, the result of asking the fundamental question, *how can we generate sentences for augmentation without changing their true labels?* We do not expect EDA to be the go-to augmentation method for NLP, either now or in the future. Rather, we hope that our line of thought might inspire new approaches for universal or task-specific data augmentation.

Now, let's note many of EDA's limitations. Foremost, performance gain can be marginal when data is sufficient; for our five classification tasks, the average performance gain for was less than 1% when training with full datasets. And while performance gains seem clear for small datasets, EDA might not yield substantial improvements when using pre-trained models. One study found that EDA's improvement was negligible when using ULMFit (Shleifer, 2019), and we expect similar results for ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). Finally, although we evaluate on five benchmark datasets, other studies on data augmentation in NLP use different models and datasets, and so fair comparison with related work is highly non-trivial.

7 Conclusions

We have shown that simple data augmentation operations can boost performance on text classification tasks. Although improvement is at times marginal, EDA substantially boosts performance and reduces overfitting when training on smaller datasets. Continued work on this topic could explore the theoretical underpinning of the EDA operations. We hope that EDA's simplicity makes a compelling case for further thought.

8 Acknowledgements

We thank Chengyu Huang, Fei Xing, and Yifang Wei for help with study design and paper revisions, and Chunxiao Zhou for insightful feedback. Jason Wei thanks Eugene Santos for inspiration.

[1] (Fadaee et al., 2017) for translation
[2] (Yu et al., 2018) for comprehension
[3] (Hu et al., 2017) for text classification
[4] (Xie et al., 2017) for translation
[5] (Sennrich et al., 2016) for translation
[6] (Kolomiyets et al., 2011) for temporal analysis
[7] (Kobayashi, 2018) for text classification
[8] (Wang and Yang, 2015) for tweet classification
[9] EDA does use a synonym dictionary, WordNet, but the cost of downloading it is far less than training a model on an external dataset, so we don't count it as an "external dataset."

References

Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. 2015. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(9):1469–1477.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573. Association for Computational Linguistics.

Murthy Ganapathibhotla and Bing Liu. 2008. Mining opinions in comparative sentences. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 241–248, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *ICML*.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio augmentation for speech recognition. In *INTERSPEECH*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL-HLT*.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 271–276, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2873–2879. AAAI Press.

Qian Liu, Zhiqiang Gao, Bing Liu, and Yuanlin Zhang. 2015. Automated rule selection for aspect extraction in opinion mining. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1291–1297. AAAI Press.

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

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

David Rolnick, Andreas Veit, Serge J. Belongie, and Nir Shavit. 2017. Deep learning is robust to massive label noise. *CoRR*, abs/1705.10694.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96. Association for Computational Linguistics.

Sam Shleifer. 2019. Low resource text classification with ulmfit and backtranslation. *CoRR*, abs/1903.09244.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99. Association for Computational Linguistics.

Patrice Simard, Yann LeCun, John S. Denker, and Bernard Victorri. 1998. Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of*

the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop, pages 239–27, London, UK, UK. Springer-Verlag.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Parsing With Compositional Vector Grammars. In *EMNLP*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. *CoRR*, abs/1409.4842.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. pages 1422–1432.

Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66.

Laurens Van Der Maaten. 2014. Accelerating t-sne using tree-based algorithms. *J. Mach. Learn. Res.*, 15(1):3221–3245.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563. Association for Computational Linguistics.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Levy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 649–657, Cambridge, MA, USA. MIT Press.

9 Supplementary Material

9.1 Implementation Details

All code for EDA and the experiments in this paper can be downloaded for any use or purpose: http://github.com/jasonwei20/eda_nlp. The following implementation details were omitted from the main text:

Synonym thesaurus. All synonyms for synonym replacements and random insertions were generated using WordNet (Miller, 1995).

Word embeddings. We use 300 dimensional word embeddings trained using GloVe (Pennington et al., 2014).

CNN. We use the following architecture: input layer, 1D convolutional layer of 128 filters of size 5, global 1D max pool layer, dense layer of 20 hidden units with ReLU activation function, softmax output layer. We initialize this network with random normal weights and train against the categorical cross-entropy loss function with the adam optimizer. We use early stopping with a patience of 3 epochs.

RNN. The architecture used in this paper is as follows: input layer, bi-directional hidden layer with 64 LSTM cells, dropout layer with p=0.5, bi-directional layer of 32 LSTM cells, dropout layer with p=0.5, dense layer of 20 hidden units with ReLU activation, softmax output layer. We initialize this network with random normal weights and train against the categorical cross-entropy loss function with the adam optimizer.We use early stopping with a patience of 3 epochs.

9.2 Benchmark Datasets

Summary statistics for the five datasets used are shown in Table 5.

Dataset	c	l	N_{train}	N_{test}	$\|V\|$
SST-2	2	17	7,447	1,752	15,708
CR	2	18	4,082	452	6,386
SUBJ	2	21	9,000	1,000	22,329
TREC	6	9	5,452	500	8,263
PC	2	7	39,418	4,508	11,518

Table 5: Summary statistics for five text classification datasets. c: number of classes. l: average sentence length (number of words). N_{train}: number of training samples. N_{test}: number of testing samples. $|V|$: size of vocabulary.

10 Frequently Asked Questions

FAQ on implementation, usage, and theory.

10.1 Implementation

Where can I find code? http://github.com/jasonwei20/eda_nlp

How do you find synonyms for synonym replacement? We use WordNet (Miller, 1995) as a synonym dictionary. It is easy to download.

Is there an EDA implementation for Chinese or other languages? Not yet, but the implementation is simple and we encourage you to write your own and share it.

10.2 Usage

Should I use EDA for large datasets? Similar to how in vision, adding color jittering might not help when you're training a classifier with a large number of images, EDA might not help much if you're using a large enough dataset.

Should I use EDA if I'm using a pre-trained model such as BERT or ELMo? Models that have been pre-trained on massive datasets probably don't need EDA.

Why should I use EDA instead of other techniques such as contextual augmentation, noising, GAN, or back-translation? All of the above are valid techniques for data augmentation, and we encourage you to try them, as they may actually work better than EDA, depending on the dataset. But because these techniques require the use of a deep learning model in itself to generate augmented sentences, there is often a high cost of implementing these techniques relative to the expected performance gain. With EDA, we aim to provide a set of simple techniques that are generalizable to a range of NLP tasks.

Is there a chance that using EDA will actually hurt my performance? Considering our results across five classification tasks, it's unlikely but there's always a chance. It's possible that one of the EDA operations can change the class of some augmented sentences and create mislabeled data. But even so, "deep learning is robust to massive label noise" (Rolnick et al., 2017).

10.3 Theory

***How* does using EDA improve text classification performance?** Although it is hard to identify exactly how EDA improves the performance of classifiers, we believe there are two main reasons. The first is that generating augmented data similar to original data introduces some degree of noise that helps prevent overfitting. The second is that using EDA can introduce new vocabulary through the synonym replacement and random insertion operations, allowing models to generalize to words in the test set that were not in the training set. Both these effects are more pronounced for smaller datasets.

It doesn't intuitively make sense to make random swaps, insertions, or deletions. How can this possibly make sense? Swapping two words in a sentence will probably generate an augmented sentence that doesn't make sense to humans, but it will retain most of its original words and their positions with some added noise, which can be useful for preventing overfitting.

For random insertions, why do you only insert words that are synonyms, as opposed to inserting any random words? Data augmentation operations should not change the true label of a sentence, as that would introduce unnecessary noise into the data. Inserting a synonym of a word in a sentence, opposed to a random word, is more likely to be relevant to the context and retain the original label of the sentence.