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.