Easy Data Augmentation Techniques for Traditional Machine Learning Models on Text Classification Tasks

Lukas Santing

5727456

Utrecht University

Bachelor Artificial Intelligence

7.5 EC

16-4-2021

First Supervisor

Dr. Dong Nguyen

Second Supervisor

Dr. Jakub Dotlacil

Contents

Abstract 2
1. Introduction 3
2. Background 4
2.1 Data Augmentation 4
2.2 Easy Data Augmentation 5
2.3 Neural Networks & Traditional Machine Learning Models 7
2.4 Logistic Regression 7
2.5 Naïve Bayes 8
2.6 Decision Tree 8
3. Method 8
3.1 Experimental Setup 8
3.2 Datasets 9
3.3 Pre-processing 10
3.4 Easy Data Augmentation 10
3.5 Machine learning models 12
3.6 Evaluation metrics 12
4. Results 12
4.1 Performance per classifier 12
4.2 Performance per dataset 13
5. General Discussion 15
6. Conclusion 16
7. References 16

Abstract

The use of data augmentation techniques in NLP for the creation of more robust models has increased in recent years. Easy Data Augmentation (EDA) techniques by Wei & Zou (2019) proposed a simple method to augment small datasets for text classification that showed promising results. While most research in the topic of data augmentation for NLP has been focused on deep learning models and not traditional machine learning models, these models are still commonly used for text classification. On three text classification tasks, this research tests the application of EDA on the performance of three traditional machine learning models: logistic regression, naïve bayes and decision tree. Results show that EDA marginally improves performance for these classifiers on small and large datasets.

1. Introduction

Natural Language Processing (NLP) is a key area of Artificial Intelligence (AI), and text classification is a fundamental task in NLP. Text classification in NLP consists of the task of automatically assigning one or more classes to a text or document. The accurate classification of texts is a challenge relevant to many fields.

Training a machine learning model to perform accurately on a given task requires large sets of relevant data for it to learn from, especially for deep learning models such as neural networks. Datasets are typically divided into training and testing sets, where, as the names suggest, training sets are used to train a learning model and testing sets are used to test the performance of the model after training. Datasets of sufficiently large sizes to allow for adequate training are not always available or easy to collect. Both deep learning models, such as neural networks, and traditional machine learning models, such as logistic regression classifiers, have shown improvements in performance on text classification tasks, though such performance is dependent on sufficient amounts of data.

To tackle the problem of sufficient data for text classification Wei and Zou (2019) developed easy data augmentation (EDA) techniques to boost performance on text classification tasks by the application of four simple operations such as replacement of words with their synonyms that could be easily applied to sentences within the text and have relatively no cost. The application of these operations to the training set allows for an expansion of existing data and provides a larger training set to learn a classifier on. On five tasks, both convolutional and recurrent neural networks showed improvement in accuracy while only using 50% of the available training data, showing that the use of EDA can show improvement for neural networks on text classification tasks.

Deep learning models, however, have significant disadvantages. Though they tend to outstrip classic machine learning models on performance, they are also more expensive with respect to time and computation power needed to train models capable of yielding such results. They further provide less insight into the reasoning behind a decision made by the model than traditional machine learning models, information that can be crucial. Effective neural networks come at a cost, and if the requirements cannot be met then alternatives such as traditional machine learning models must be considered. EDA techniques to improve performance on text classification tasks could also be applied to simpler machine learning models that do not have the disadvantages of deep learning models, but could still benefit from data augmentation.

Traditional machine learning models such as naïve bayes classifiers have commonly been used for text classification (Chen et al., 2009), as have decision tree and logistic regression classifiers (Pranckevičius et al, 2017). They require significantly less time to train when compared deep learning models and do not require better hardware for more complex operations. Considering the significant

improvements in performance that EDA techniques show for deep learning models on little data, a relevant question is whether similar improvements can be found for traditional machine learning models. If such improvements of relevant significance can be found, then smaller datasets could increasingly be used in text classification tasks without the requirement for complex deep learning models.

This leads me to my research question: To what extent can easy data augmentation techniques improve the performance of traditional machine learning models on text classification tasks?

In order to test this, I will train a naïve bayes, a decision trees and a logistic regression classifier on three different text classification datasets and compare their performance on data that was modified by EDA techniques against performance when the data was not modified. Subsets from each dataset of various sizes will be used in each case to analyse the performance of the models and the effects of EDA on various amounts of data.

The structure of this thesis is as follows: Section 2, the background, will discuss both the relevant work related to this research and the important concepts and tools described in this thesis. Section 3 will discuss the experimental setup, namely how all experiments are carried out. The results of these experiments will be presented in section 4. How these results relate to the research question will be reflected on in the general discussion in section 5, which will be followed by the conclusion in section 6.

2. Background

2.1 Data Augmentation

In order to train a machine learning model of any type to adequately perform a given task, datasets of sufficient size and quality are needed to train and test the model. The task of learning a model on a dataset that is relatively small thus poses a significant problem. Data augmentation offers a potential solution to this problem. Data augmentation expands the given dataset by creating new, slightly modified versions of existing data samples. This allows for small datasets to artificially grow by the creation of new data samples based on the original dataset.

Automatic data augmentation techniques are commonly used to help train better models when using small datasets in a variety of fields. Especially within the computer vision branch of machine learning, state of the art models increasingly use augmentation techniques on images (Krizhevsky et al., 2017). The use of a single image in a dataset, for example, can be expanded by randomly rotating it, cropping it, flipping it on its axes, or with the application of grayscale and smoothing filters. Relatively small datasets can be artificially increased using such methods. It has also been effectively used in speech tasks by Xiaodong Cui et al. (2015) by using techniques such as vocal tract length perturbation. This

is a technique that supplements the dataset by artificially changing the vocal tract length of speech samples, thus creating slightly augmented data samples.

The field of NLP is no longer a stranger to data augmentation, which in recent years has seen the use of multiple creative and complex techniques to augment data. Yu et al. (2018) employed back-translation as a data augmentation technique. This method translates texts into another language, such as French, and then back into English in order to obtain paraphrases of those texts. Similarly, Fadee et al. (2017) uses translation and substitution as data augmentation. Here, the data to be expanded is presented in the form of pairs of sentences in their original (source) and translated (target) form. Data is augmented by replacing words in the target sentences with translations of substitutions of words in their source sentence. This creates new source and target sentence pairs by the use of substitution and translation as augmentation. Another method of data augmentation in NLP uses data noising methods as smoothing (Xie et al., 2017). Their goal was to tackle the problem of overfitting caused by having little data, and so they augmented data using various noising techniques such as unigram noising or blank noising.

The main drawback that all of these techniques that explore data augmentation in NLP have in common is that they have a significant cost relative to any performance gain. Ideally, data augmentation techniques would be simple enough to not sacrifice the efficiency of a model, as is reflected in the conceptual simplicity of the computer vision data augmentation techniques mentioned previously. Various methods that apply simple data augmentation techniques directly within the text have been researched, such as Abulaish & Sah (2019) where positive and negative documents in a sentiment analysis corpus were supplemented with positive and negative phrases, respectively. However, the work of Wei and Zou (2019) in their paper introducing EDA pioneers a conceptually simple but robust method for the application of data augmentation techniques directly to text classification tasks, through the use of four operations that can be applied directly within a text.

2.2 Easy Data Augmentation

Wei and Zou (2019) developed a set of operations that served as data augmentation techniques for text classification that work with relatively no cost.

Given a sentence in the training set, one of the following operations is randomly chosen and applied:

- Synonym Replacement (SR), where a random word in the sentence is swapped with a random synonym,
- Random Insertion (RI), where a random synonym of a random word in the sentence is inserted into the sentence,
- Random Swap (RS), where random words within the sentence are swapped, and
- Random Deletion (RD), where a random word within the sentence is deleted.

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 as used by Wei & Zou (2019) with a sample sentence from Socher et al. (2013).

The number of words that can be changed within a sentence without adding too much noise is dependent on the length of the sentence, so longer sentences can absorb more noise and suffer more augmentation. The number of augmented sentences that is generated per original sentence can also easily be changed.

An important question concerning EDA is whether true labels are conserved. To test this, Wei and Zou (2019) applied t-SNE, an embedding technique commonly used for the visualization of high-dimensional data in a scatter plot (Van Der Maaten, 2014), to results from a recurrent neural network (RNN) trained on augmented data and the same RNN trained on non-augmented data. The 2-D representations of this data showed that the representations for augmented sentences clustered closely to the original sentences of the same class, suggesting that EDA augmented sentences conserve the labels of the original sentences. Considering that the implementation of EDA is not changed from the original for this research, the same test is not repeated in this paper.

Wei and Zou (2019) tested their operations on five benchmark datasets with both convolutional and recurrent neural networks. They analysed the results of the application of EDA when the training data had been resized to 500, 1000 and 5000 samples, alongside a full dataset. Results were encouraging, and showed that EDA application resulted in improvements in accuracy ranging from 0.8 to 3.8% averaged across all 5 datasets. When applied to datasets of 500 samples, the amount of improvement tended to be higher than on the full-sized dataset, showing that especially when applied to limited datasets, EDA for neural networks can show improvement in performance.

However, regarding EDA and the related work in NLP as discussed in section 2.1, all of these examples of data augmentation for text classification are tested on deep learning models. Work has been done concerning EDA, for example Jang et al. (2020) that present work on a sequential targeting (ST) method to address imbalance in text data, showing that their implementation of ST with EDA gave high improvement scores with a convolutional neural network. Research by Marivate and Sefara (2020, p. 395) analysed the effect of various data augmentation techniques such as round-trip translation and synonym replacement on the performance of deep neural network models compared against a logistic regression classifier and showed promising reductions in error, though they did not

use EDA. To my knowledge, little other research has been done to show the effectiveness of data augmentation techniques in NLP for traditional machine learning models.

2.3 Neural Networks & Traditional Machine Learning Models

Deep learning models such as deep neural networks are a type of machine learning model with a complex multi-layered structure. These models do feature extraction and classification automatically with little required intervention. Their complex multi-layered structure requires a relatively large dataset, typically of millions of samples, to eliminate fluctuations and make accurate predictions. This complex nature is also the reason that deep learning models can require weeks for training, as well as powerful computers capable of running complex computations. While deep learning is still in its infancy in some areas when compared to non-deep learning models, it tends to outstrip non-deep learning models in performance, and its enormous power has led it to being used in state-of-the-art systems in many areas, especially when its relatively costly demands can be met.

Non-deep learning models such as logistic regression, naïve bayes and decision tree classifiers are also called traditional machine learning models. Models such as these were the classic solution to many machine learning problems before the rise of deep learning, and many are still used today. When it comes to required time and computing power these models tend to be cheaper than deep learning models, and as such offer a simpler "off the shelf" solution to problems such as text classification. Another advantage offered is that their simpler and easier to understand structure requires much less data than deep learning models, tending towards thousands of data samples instead of millions. However, these models do not perform feature engineering automatically as neural networks do, and so arguably require the researcher to have a deeper understanding of the subject in question.

The three traditional machine learning models relevant to this research are the logistic regression, naïve bayes and decision trees classifiers.

2.4 Logistic Regression

Contrary to what a name with 'regression' may suggest, logistic regression is used for classification. It uses a logistic function to produce discrete binary outputs. It makes the central assumption that $P(Y|X)$ can be approximated as a sigmoid function applied to a linear combination of input features (Jurafsky & Martin, 2009, pp. 231–234):

$$\mathrm{P(Y|X)} = \frac{1}{1+\mathrm{e}^{-\mathrm{f(x)}}} \quad (1)$$

where $f(x)$ is a function consisting of features (x) and their weights (β):

$$\mathrm{f(x)} = \mathrm{x}_0 + \mathrm{x}_1\beta_1 + \cdots + \mathrm{x}_\mathrm{k}\beta_\mathrm{k} + \varepsilon \quad (2)$$

where $x, \beta, f(x) \epsilon R^k$ and ε represents noise in the data generating process. Using the posterior probability function, the estimation function $f(x)$ can be rewritten as the 'log of odds' ratio:

$$\log \left[\frac{P(Y|X)}{1-P(Y|X)} \right] = x_0 + x_1\beta_1 + \cdots + x_k\beta_k + \varepsilon = f(x) \quad (3)$$

The logarithmic transformation helps the learning function by normalizing values and ensuring no features dominate the outcome. Given a set of features, the observation can be classified as 'true' (logistic regression is used for binary classification) if the function in (2) is greater than 0. The logistic regression function can thus be seen as learning a hyperplane that separates points in space that are in the 'true' class from those that are not. (Jurafsky & Martin, 2009, pp. 231–234).

2.5 Naïve Bayes

The naïve bayes classifier is quite intuitive. As the name suggests, is based on Bayes theorem with the (naïve) assumption that each value (feature) is independent from the rest:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)} \quad (4)$$

As $(x|1, \ldots, x_n)$ is constant given the input, the classification rule can be as follows (Scikit-learn, 2011):

$$\hat{y} = \underset{y}{\operatorname{argmax}} P(y) \prod_{i=1}^{n} P(x_i|y) \quad (5)$$

2.6 Decision Tree

The decision tree classifier is a simple machine learning model that uses the idea of transferring data to a decision tree diagram and can be used for both regression and classification purposes. It consists of (1) decision nodes where a decision must be made concerning a feature of the data, (2) edges as answers from a node to build a connection to the next nodes, and (3) leaf nodes which are outcomes for classification. Features in the data are split and are represented in this format as the decision between different nodes in the tree. During classification, the class that eventually gets landed on in the leaf nodes is decided as the prediction.

The advantages of decision tree classifiers is their simplicity, and how little data preparation is necessary. A notable disadvantage of using decision trees for classification is that they can overfit the data and create overly complex trees that do not generalise the data well (Scikit-learn, 2011).

3. Method

3.1 Experimental Setup

In this research I explore the effects of EDA techniques as developed by Wei and Zou (2019) on the performance of various traditional machine learning classifiers when trained and tested on three

different text classification datasets. The datasets include no pre-defined splits and are separated into three sets using a 60/20/20 (%) train, development, and test split. While many modern datasets with millions of samples will split at 95% or higher for training, the datasets used in this research are relatively small at only about 10,000 to 30,000 samples, so the classic 60% is sufficient for training and leaves adequately sized sets for both development and testing. The development set is required to find the optimal parameter settings for EDA and to prevent optimizing on the test set.

Following the split, a random stratified subset is selected from the training set equal to N = 500, 1000, 5000, to compare the effects of EDA on datasets of various sizes. Both the development and testing sets are left at their full size. The next step is the application of EDA to the training set, the specifics of which are discussed in section 3.4. These provide the final instances for training. In each case, the classifiers are trained on both a nonmodified version of the dataset and a version that has been modified and expanded using EDA.

3.2 Datasets

For this research, three benchmark datasets were chosen. The datasets used are (1) **SUBJ**: The Subjectivity / Objectivity dataset from Pang & Lee (2004), (2) **CLICKB**: The Clickbait data set from Chakraborty et. al (2016), and (3) **HATESP**: The Hate-Speech / Offensive language dataset from Davidson et. al (2017). The first dataset was used by Wei & Zou (2019) in their paper on EDA. A benefit of using this dataset is that it allows for comparison of results in this paper with their results on one of their chosen datasets. All three datasets were chosen as they have a different subject matter from the rest, to allow a broad range of subject matter to be expanded by EDA.

The SUBJ dataset contains a list of 10,000 sentences from a collection of movie reviews from IMDB. 5000 of these are labelled as objective, 5000 as subjective. An example of an 'objective' sentence from this dataset would be "*believing his parent's dead, bruce is raised by adoptive parents*" while a 'subjective' sentence would be "*vile and tacky are the two best adjectives to describe ghost ship*."

The CLICKB dataset contains a list a total of 32,000 news headlines that are listed as either clickbait or not clickbait. The authors automatically collected articles from Wikinews as non-clickbait headlines due to the style guides and rigorous checks employed by Wikinews for their articles, while clickbait articles were selected from less trustworthy headlines. This was done manually by volunteers to avoid false negatives. The dataset is split equally between the two, each class containing 16,000 samples. Each of these titles is considered a 'sentence' for the purposes of EDA. An example of a clickbait sentence is "*Natalie Dormer And Sam Claflin Play A Game To See How They'd Actually Last In 'The Hunger Games*'" while a non-clickbait sentence would be "*Coldplay's new album hits stores worldwide this week*".

The HATESP dataset is a collection of 24,783 tweets that are labelled as containing either hate-speech, offensive language, or neither. This data was chosen as a random subset of tweets containing words identified as hate speech on *Hatebase.org* collected automatically from twitter. These were later manually split into the three categories. The data split is not equal, with 1431 counts of hate-speech, 19190 counts of offensive language and the remaining 4162 samples counting under neither. An example of a sentence with hate speech is *"#WestVirginia is full of white trash*.", while a sentence with offensive language is ""*All these b****** want a baby, I don't want no children*."" (censorship by me), and an example of neither is "*The #Yankees don't need Tanaka or Beltran or all these 'big name' players!!! #believe*". For the purpose of EDA, each tweet is considered a 'sentence'.

3.3 Pre-processing

The datasets require some pre-processing before the models can be trained. The first step is to load them into 'Pandas' data frames (McKinney, 2010) for manipulation, upon which all stop words are removed from the data. The list of stop words used is the same as those listed in the original EDA code from the open GitHub repo of Wei and Zou (2019). Any words in the dataset appearing in this list are removed before proceeding to the splitting, EDA application, and training steps.

Using the pre-processing tools from SciKit learn (Scikit-learn, 2011), the sentences in the training, development and test sets are vectorized using the Tf-Idf vectorizer and count vectorizer included in the SciKit Learn package, in order to have a training set with features that could be used by the models (Scikit-learn, 2011).

3.4 Easy Data Augmentation

The EDA operations applied are the same as those as implemented by Wei & Zou (2019) and as described in section 2.2. Each of these four operations can be edited as a parameter with a value between 0 and 1, describing the percentage of words that will be changed in the sentence according to that rule. If, for example, I wanted to replace 3% of words with their synonyms I would set the SR value to 0.3. The number of operations applied within a sentence is dependent on the length of the sentence, as longer sentences can absorb more 'noise' than shorter sentences without losing their class label. Besides these four operations, the fifth parameter that could be edited is the number of augmented sentences generated per original sentence.

The recommended EDA parameter settings from Wei and Zou (2019) are as follows:

N_{train}	α	n_{aug}
500	0.05	16
2000	0.05	8
5000	0.1	4
More	0.1	4

Table 2: Wei & Zou (2019) EDA recommended usage parameters. N_{train} stands for the number of training samples, α stands for the parameter settings and n_{aug} stands for the number of augmented sentences to create for each sentence.

Finding the ideal balance of settings for each dataset is a delicate and quite sensitive task, and the development set is vital to fine tune these parameter balances as much as possible before finally testing on the test set. The default parameter settings in the code from the GitHub repository of Wei and Zou (2019) differ slightly from the recommended parameters as shown in Table 2. In the default Python code parameters, each of the four operations parameter is set to 0.1 and the number of augmented sentences per original was set to 9. This means that 10% of each sentence are subject to change by each operation, and that the size of a training set would increase by 10 after application of EDA. To find the ideal base parameters, initial parameter settings are set to the code default, and are tested against the recommended settings as listed in table 2. There seems to be no significant difference in performance between these two settings.

Optimal parameter values are found by testing various settings over the development set at each subset training size and comparing values from training on sets not modified with EDA against sets that are modified with various parameter values. Relying on the accuracy and F1-score as evaluation metrics then allows for the fine-tuning of these parameter values.

Accomplishing the task of finding the right EDA parameters requires the testing of certain expectations. For example, my expectation of the hate-speech and offensive language (and neutral) dataset would be that the line between the three classes is quite thin with regards to the specific words used, and so the replacement of words with their synonyms with EDA could be more likely to introduce noise by not preserving their class label.

Testing this hypothesis, however, shows that a higher synonym replacement tends to yield better results in each case. On reflection, a synonym replacement function that results in incorrectly labelled sentences would not be achieving its purpose, so this result shows two things: firstly, that the function performs its task reliably, and secondly, that synonym replacement is an effective function in text data augmentation.

In most cases, results gained using the standard settings provide a reliable indication of the expected optimal score. Within the application of this research, there is very little extra improvement to be gained with the fine tuning of EDA parameters. On average, slightly raising the parameters from their default showed little difference. Extreme values such as setting RD to 0 and setting number of augmented sentences to values greater than 9 occasionally show significant increases in performance, however these tend to be balanced with equally significant decreases, due to the random nature of the application of EDA when taken to the extremes.

One change that can be relied on however, is setting the SR parameter to high values, occasionally as high as 0.9. Though benefits from this are small and only increase up to about 0.5% on the performance of the standard parameters, these improvements are consistent.

3.5 Machine learning models

The three machine learning classifiers used in this research are **LREG**: logistic regression, **NBAYES**: multinomial naïve bayes, and **DTREES**: decision tree classifier models from the SciKit Learn package (Scikit-learn, 2011).

The logistic regression classifier is based on the logistic function modified for classification, and is implemented with l2 regularization to reduce overfitting, implemented with the standard settings.

The naïve bayes classifier is based on the application of Bayes' theorem with the assumption of independence between features. The reason for the choice of the multinomial model is that it is ideally suited for the classification of discrete features, such as word counts for text classification.

The third and final model is the decision tree classifier model, another commonly used machine learning model in text classification.

3.6 Evaluation metrics

Following the training of the classifiers, the next step is test them. Results of the classifiers over the datasets are compared on basis of their accuracy, precision, recall and F1 score.

Accuracy is simply the measure of all correctly identified cases. While it can be a good indicator of performance, if a dataset is unequal with respect to class distribution it does not provide a good representation of the effectiveness of the model.

The inclusion of F1 score allows for a more reliable result. The F1 score is the weighted average of precision and recall. Precision is the measure of correctly identified positive cases from all the predicted positive cases while recall is the measure of correctly identified positive cases from all actual positive cases. For this reason, the results of the models are presented according to their performance along the F1 score metric.

4. Results

In this section, I test EDA on three NLP tasks with a logistic regression, naïve bayes and decision tree model. Results in each case are the average of five individual measurements, due to the random nature of EDA.

4.1 Performance per classifier

All three models are run with and without EDA across all three datasets with various training set sizes. Average F1 score is shown in Table 3, with the improvements (%) printed in bold in each case. For ease of legibility, improvements are shown as percentages out of 100.

	Training Set Size			
Model	**500**	**1000**	**5000**	**Full set**
Logistic Regression	0.844	0.8673	0.899	0.9155
+ EDA	0.8483 **(+ 0.43)**	0.8825 **(+ 1.52)**	0.9117 **(+ 1.27)**	0.9225 **(+ 0.7)**
Naïve Bayes	0.8274	0.8479	0.8772	0.8849
+ EDA	0.8435 **(+ 1.61)**	0.864 **(+ 1.61)**	0.8963 **(+ 1.91)**	0.902 **(+ 1.71)**
Decision Trees	0.8138	0.8243	0.8586	0.8646
+ EDA	0.8415 **(+ 2.77)**	0.8364 **(+ 1.21)**	0.8626 **(+ 0.4)**	0.8677 **(+ 0.31)**

Table 3: Average F1 score across three text classification tasks for models with and without EDA on different training sizes

As can be seen in table 3, EDA shows some improvement, ranging from 0.31% to 2.77% increase in F1 score over results without EDA. While each case shows some sign of improvement with the use of EDA, the patterns of improvement shown are different for each classifier.

The logistic regression classifier shows more improvement on larger datasets than on smaller datasets. On a training set with 500 instances, the improvement in performance gained on EDA was only 0.43%. On training sets of size N = 1000 and 5000 however, the improvement shown was approximately three times that. The pattern does not continue at the full-size training set, however an increase of 0.7% is still shown at full size. This pattern suggests that the logistic regression classifier may plateau at certain larger datasets, with less room for improvement through EDA.

The improvement shown for the decision tree classifier shows the opposite pattern for improvement than logistic regression. Improvement was higher for smaller training sets showing the highest recorded improvement with EDA increasing performance by 2.77% at a subset of size 500. The amount of improvement decreases as the size of the training set grows, showing that EDA had more effect on smaller datasets.

The naïve bayes classifier showed another pattern of improvement entirely different from the other two classifiers, with a consistent relatively high rate of improvement across each different size of training set, with the lowest result showing at 1.61% improvement, which slightly increased at larger sized datasets.

4.2 Performance per dataset

Noteworthy results are also obtained from looking at the F1 score when averaged over all three classifiers for each dataset, as shown in table 4:

	Training Set Size			
Model	**500**	**1000**	**5000**	**Full set**
SUBJ	0.7836	0.81	0.8487	0.8513
+ EDA	0.7928 **(+ 0.92)**	0.816 **(+ 0.6)**	0.8534 **(+ 0.47)**	0.8568 **(+ 0.55)**
CLICKB	0.9177	0.9275	0.9512	0.964
+ EDA	0.9224 **(+ 0.47)**	0.938 **(+ 1.05)**	0.9575 **(+ 0.63)**	0.9665 **(+ 0.25)**
HATESP	0.7839	0.8021	0.8348	0.8497
+ EDA	0.8003 **(+ 1.64)**	0.8288 **(+ 2.67)**	0.8597 **(+ 2.49)**	0.8689 **(+ 1.92)**

Table 4: Average F1 score across three classifiers for models with and without EDA on different training sizes.

As can be seen from table 4, when averaged over three classifiers, EDA shows improvement for text classification for all three datasets in each case.

The dataset that showed the most improvement was the HATESP dataset, consistently scoring relatively high, its lowest performance increase still higher than the highest improvement of either of the other two datasets.

The F1 scores shown for the CLICKB dataset with and without EDA are already quite high. Perhaps this explains why the increase in performance seems to show increase for higher datasets, then plateaus at a certain point where there may be less room or improvement.

The scores for the SUBJ dataset show more performance increase through EDA at smaller datasets, which decreases as size of the training datasets grow. This suggests that for this dataset, more improvement is possible with less data, and that providing more data for EDA does not necessarily result in a better EDA performance. This dataset is of particular interest as it was also used by Wei & Zou (2019) for their paper on EDA. For this reason, detailed results showing the performance of each classifier on SUBJ are shown in graphs in figure 1:

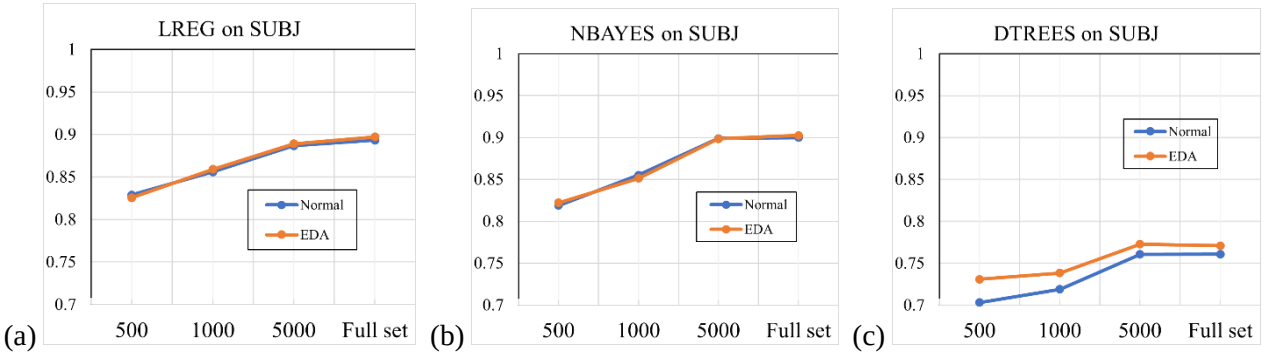

Fig. 1. F1- score of Logistic Regression, Naïve Bayes, and Decision Trees classifier on the SUBJ dataset over various training sizes.

As apparent in Figure 1 (a) and (b) both logistic regression and naïve bayes perform almost identically on the SUBJ dataset, and EDA performs very similarly in both cases, showing barely any difference, neither a significant increase nor decrease. As apparent from image (c), on this dataset the decision tree classifier performs significantly worse than both other classifiers, yet EDA consistently shows a substantial increase, with more improvement shown over smaller datasets. This suggests that for decision trees, there is more room for improvement through EDA. However, as both logistic regression and naïve bayes score higher without EDA than decision tree does with it, the choice for either of these above decision tree still seems preferable if such a choice is possible.

Comparing this to the results achieved by Wei & Zou (2019) using both convolutional and recurrent neural networks showed that their best (accuracy) score on this dataset was approximately 0.9 at the full-size dataset, and at smaller datasets scored in the low to mid-0.8 range, almost equivalent to the F1-score seen here with logistic Regression and naïve Bayes. Similarly, their EDA showed only

marginal improvement on this dataset (especially at full size), so these results are as expected. Considering that the results for the accuracy score are almost identical to F1-score for these cases as analysed in this research, these results are not presented separately.

5. General Discussion

This paper aimed to explore the effects of EDA techniques as developed by Wei and Zou (2019) on traditional machine learning classifiers for text classification tasks. This was done by applying EDA to training sets of various sizes from three different text classification tasks and training three models, namely logistic regression, naïve bayes, and decision tree classifiers, on both the augmented and non-augmented data.

EDA shows varying results when applied to different datasets and used with various classifiers. Averaged over these, marginal improvement is seen through the application of EDA. While the increase in performance seen is not dramatic, each model responds quite differently and some models, such as the naïve bayes classifier, seem to be more susceptible to these techniques than others. As discussed in section 2.3, traditional machine learning models tend to require relatively small amounts of data in order to show adequate performance. It is possible that the margin for improvement for these classifiers through data augmentation techniques is small, and that whatever improvement is possible is achieved.

The improvement seen through the application of EDA on datasets for the traditional machine learning models used in this research does not seem to be generally substantial, though the results of each classifier must also be considered separately from the others. While each classifier shows improvement across the board, the patterns of improvement are different for each of the three classifiers. Any research that intends to use EDA as way of enhancing performance of traditional machine learning models over small datasets should take these patterns into account. For example, when using EDA, the size of the dataset seems to be relevant for the choice of classifier. Considering the results of each classifier when averaged over all three datasets, when using little data (i.e., datasets with less than 1000 samples), the use of a decision tree classifier over data augmented with EDA seems to be more effective than the use of a logistic regression classifier. When using larger datasets however, the decision tree classifier loses its relative effectivity, and the naïve bayes model should be considered.

Notably, much recent research in NLP has been in the application of data augmentation techniques with the specific purposes of improving the performance of deep learning models. There is little research to be found that explores the effectiveness of such techniques for non-deep learning models that are still commonly used. This research seeks to ask that very question, though is not without its limitations. The results shown were marginal at best and showed different patterns of improvement

over the three classifiers explored. More time is needed to explore these patterns of improvement for each of these classifiers, specifically how they perform on other benchmark datasets when modified by EDA. Are these patterns of improvement seen for these other datasets? Further research could also analyse the effects of EDA techniques on the many commonly used traditional machine learning models not explored in this research, such as support vector machines. Deeper analysis can also be done to find the link (if any) between specific EDA operations and text classification tasks. To what extent does the application of specific operations provide data samples that can be effectively used to train a model for each specific task such as sentiment analysis, and to what extent do these new data samples hinder progress by introducing noise? In depth answers to questions such as these could be relevant to finding the ideal combination of EDA parameter settings for use with traditional machine classifiers, but also by extension to deep learning models training over the same datasets.

6. Conclusion

In this thesis, I have shown that the application of the EDA techniques developed by Wei and Zou (2019) show marginal improvement for traditional machine learning models for text classification in NLP. Though increases in performance are not substantial, it is a cost-effective method to provide consistent improvements across datasets of various sizes. Continued work on this topic could explore the patterns of improvement shown as well as the effects of EDA on other traditional machine learning models.

7. References

Abulaish, M., & Sah, A. K. (2019). A Text Data Augmentation Approach for Improving the Performance of CNN. *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, 625–630. https://doi.org/10.1109/comsnets.2019.8711054

Chakraborty, A., Paranjape, B., Kakarla, S., & Ganguly, N. (2016, August). Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM)* (pp. 9-16). IEEE.

Chen, J., Huang, H., Tian, S., & Qu, Y. (2009). Feature selection for text classification with Naïve Bayes. *Expert Systems with Applications*, *36*(3), 5432–5435. https://doi.org/10.1016/j.eswa.2008.06.054

Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017, May). Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 11, No. 1).

Fadaee, C. (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)* (pp. 567–573). Association for Computational Linguistics.

Jang, J., Kim, Y., Choi, K., & Suh, S. (2020). Sequential Targeting: an incremental learning approach for data imbalance in text classification. *arXiv preprint arXiv:2011.10216*.

Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing* (2nd ed.). Prentice Hall.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Marivate, V., & Sefara, T. (2020). Improving Short Text Classification Through Global Augmentation Methods. *Lecture Notes in Computer Science*, 385–399. https://doi.org/10.1007/978-3-030-57321-8_21

McKinney, W., & others. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56).

Pang, B., & Lee, L. (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* (pp. 271–es). Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research*, 12, 2825–2830.

Pranckevičius, T., & Marcinkevičius, V. (2017). Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*, *5*(2), 221–232. https://doi.org/10.22364/bjmc.2017.5.2.05

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1631-1642.

Van Der Maaten, L. (2014). Accelerating T-SNE Using Tree-Based Algorithms. *J. Mach. Learn. Res., 15(1), 3221–3245.*

Wei, J. & Zou, K. (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) (pp. 6382–6388). Association for Computational Linguistics.

Xiaodong C., Goel, V., & Kingsbury, B. (2015). Data Augmentation for Deep Neural Network Acoustic Modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *23*(9), 1469–1477. https://doi.org/10.1109/taslp.2015.2438544

Xie, Z., Wang, S. I., Li, J., Lévy, D., Nie, A., Jurafsky, D., & Ng, A. Y. (2019*)*. Data noising as smoothing in neural network language models. Paper presented at 5th International Conference on Learning Representations, ICLR 2017, Toulon, France.

Yu, A. W., Dohan, D. Le, Q. V., Luong, M. T., Zhao, R., & Chen, K. (2018). Fast and Accurate Reading Comprehension by Combining Self-Attention and Convolution. In *International Conference on Learning Representations.*