Analyzing Easy Data Augmentation Techniques for Text Classification

UNDERGRADUATE THESIS

Analyzing Easy Data Augmentation Techniques for Text Classification

---

Author:
Carolyn Wong

Presented to the Department of Computer Science
in partial fulfillment of the requirements for an A.B. degree with Honors

Harvard College
March 2021

Abstract

In natural language processing, text classification is the task of assigning a category to a given text example. Text classification has a variety of applications ranging from automated processing of customer reviews to spam detection. Current state-of-the-art approaches for text classification tasks use neural language models. These models are resource-intensive, requiring large amounts of labeled training data. However, training data may not always be available in large quantities, especially for low-resource languages, and labeled data is often laborious to obtain. Consequently, it is desirable to understand the factors contributing to text classification models' performance. I address several questions about which factors contribute to the high performance achieved by the current state-of-the-art neural models. To do so, I analyze traditional and neural methods for a diverse range of text classification tasks. I study various properties such as model assumptions and word vector representations to determine the effect of each of these features on text classification performance. On the best performing models from these understandings, I evaluate existing data augmentation techniques for text classification proposed by Wei and Zou (2019), which are methods that perform simple text editing operations to generate new training examples. However, such existing data augmentation techniques require external datasets or knowledge about the semantic properties of words. To this end, I propose and assess a novel length-based method that does not require external linguistic knowledge. This method replaces words with other words of similar length, as word length closely reflects the average information content and conceptual complexity of words in English (Piantadosi, Tily, and Gibson, 2011; Lewis and Frank, 2016). I demonstrate that this length-based technique adds consistent gains for several of the evaluated text classification tasks.

Acknowledgements

First, I would like to thank my advisor Professor Stuart Shieber, for introducing me to the world of NLP research, for mentoring and supporting me throughout the various stages of this project, and for offering many helpful suggestions when I became stuck.

Additionally, I would also like to thank my two thesis readers, Professor Roger Levy, for teaching me so many fascinating things about the field of computational psycholinguistics and for fielding all of my questions in his weekly office hours, and Yuntian Deng, for his infinite patience in explaining NLP concepts to me both in and outside of class, for taking the time to read through many drafts, and for offering so much detailed feedback and guidance.

I would like to express my deepest appreciation to all of the following individuals for making my experience at Harvard an unforgettable and life-changing experience: Vivian Lee, for entertaining me with her witty jokes and dance moves; Filippos Sytilidis, for periodically sending me bird and dog pictures as encouragement; Mirac Suzgun, for inspiring me to try NLP research and advising me in the very beginning stages of this thesis; Mushtaaq Ali, for all of the pseudo-work done in Quincy dining hall and facts about birds; Raymond So, for his hilarious stories about burritos, haircuts, and Clover mushroom burgers, and for (almost) taking Stat110 with me; Jessica Ehondor, for being there from the very beginning of my journey into linguistics, for the works of art and memories made in online Pictionary, and for providing reliable advice on practically everything in life; Zach Yedidia, for the late night work sessions, encouragement, and conversations, for never failing to make me laugh, for introducing me to different genres of music, and for sending me innumerable birb videos; and finally, my wonderful housemates, Eliane Grace and Carter Nakamoto, for being a source of so much joy. I am incredibly thankful for the freshly baked brownies left just outside my door, their tolerance for my terrible puns, and the many long conversations over meals, and I am truly so grateful to spend my last few months at Harvard with them.

Lastly, I would like to express my warmest gratitude to my family: my sister Cathy, for introducing me to computer science, for providing guidance and advice on research and life, for telling me so many amazing stories, and for always reaching out to check in and listening so patiently, and my parents, for all of their lifelong support and love.

Contents

Chapter 1

Introduction

My research focuses on analyzing how data augmentation and different features of text classification models contribute to performance on text classification tasks.

Wei and Zou (2019) evaluate four "simple" data augmentation techniques: synonym replacement, random swap, random insertion, and random deletion. They show that these techniques, collectively called Easy Data Augmentation, increase the accuracy of convolutional and recurrent neural networks on five selected text classification tasks.

Despite the high performance of Easy Data Augmentation with neural models, neural models are relatively complex compared to other traditional models used for text classification. It remains unclear as to which factors contribute most significantly to the performance of the current state-of-the-art neural approaches to text classification tasks. In this work, I address several questions about the features of text classification model architectures that contribute to classification accuracy.

I investigate five tasks used to evaluate text classification models in Wei and Zou (2019), four in sentiment classification and one in question classification. Wei and Zou (2019) found that convolutional neural network (CNN) models achieved an average accuracy of 88.3% across the tasks. However, it is unclear whether it is possible to use a model lower in complexity than neural models to achieve similar accuracy performance on the text classification tasks. Therefore, to determine the overall "difficulty" of the text classification tasks, I establish a baseline model by evaluating the accuracies of various Naive Bayes classifiers. The Naive Bayes classifier is useful as a baseline because it requires far less computational power and time to train compared to a neural text classification model.

Using the Naive Bayes classifier, I obtain an average accuracy of 78.2% across the classification tasks for the highest performing variant of the Naive Bayes classifier. For these classification tasks, CNNs therefore afford a significant increase in accuracy (+10.1%) relative to a Naive Bayes classifier. This result raises the question: what aspects of the CNN contribute to its higher accuracy?

The Naive Bayes classifier differs from the CNN in that it uses one-hot vector representations for the vocabulary, in contrast to the pretrained GloVe embeddings used in the CNN

model. Additionally, the Naive Bayes classifier makes two strong assumptions about the data: the co-occurrence assumption and the word order assumption.

1. *Co-occurrence assumption: the probabilities of the features within a document are independent, given the classification for the given document.* To study the effect of the co-occurence assumption, I compare the accuracy of the unigram Naive Bayes classifier to the accuracy of a CNN using one-hot vector representations trained on the task corpora with random word order.

2. *Word order assumption: the order of words within a document does not matter.* To study the effect of the word order assumption, I compare the accuracy of the one-hot CNN trained on the task corpora with random word order to the accuracy of a one-hot CNN trained on the task corpora with original word order.

The above comparisons of the unigram multinomial Naive Bayes classifier, the one-hot CNN trained with random word order, and the one-hot CNN trained with original word order study the effects of the co-occurrence and word order assumptions. However, current state-of-the-art CNN models used for text classification also implement pretrained word embeddings to represent the vocabulary, which are not present in Naive Bayes models. These pretrained word embeddings project words into a vector space of lower dimension than the vocabulary. Consequently, the vector representations of words for the neural models are of lower dimension compared to the representations for the Naive Bayes models. I isolate and examine the effect of word embeddings' lower dimension by comparing the accuracy of a CNN trained on one-hot vocabulary representations (a high dimension representation of the vocabulary), to a CNN trained with randomly generated word embeddings (a low dimension representation of the vocabulary). I find that decreasing the dimension of word embeddings has a significantly positive effect on accuracy when averaging across the tasks. I also find that the performance gains for each individual classification task vary significantly, with modest gains for sentiment classification and the greatest performance gains for question classification. These greater performance gains for question classification compared to sentiment classification could occur because question classification may rely on the presence of a word from a well-defined set of question words (such as *who, why, where*) to determine the category. Using word vector representations with a lower dimension would allow the model to better learn the features of such keywords. In contrast, sentiment classification focuses on the meaning of words in an example; the semantic meanings of words may be less easily captured when only using representations of a lower dimension.

However, *pretrained* word embeddings such as the GloVe embeddings used in current state-of-the-art neural models are pretrained on a large external corpus of text data, allowing the embeddings to encode linguistic information such as semantic relationships between words. To analyze the effect of pretraining word embeddings, I compare the accuracy of the CNN

using random word embeddings to a CNN using pretrained GloVe word embeddings. My results demonstrate that pretraining word embeddings also has a large positive effect on accuracy when averaging across the tasks. Again, the performance gains vary among individual tasks. Greater performance gains are observed for sentiment classification compared to those for question classification. These greater performance gains may occur because sentiment classification depends on the semantic meaning of the words for a given example. The semantic meaning encoded in pretrained word embeddings would therefore contribute to the model's ability to predict sentiment. Such linguistic knowledge in pretrained word embeddings may have less of an effect for question-type classification tasks, where the semantics of words may have a smaller role in classification.

Finally, the weights of the GloVe word embeddings can also be further adjusted when training a text classification model on a task corpus, potentially allowing embeddings to encode information specific to that classification task. In a further exploration of this property, I evaluate and compare the accuracy of a CNN with GloVe word embeddings with fixed weights during training, to a CNN with GloVe word embeddings where the embedding weights are further adjusted during training on the task corpora. I find that training word embeddings on the task corpora has different effects for each classification task, but leads to small improvements in accuracy when averaging across the tasks. For some tasks, the performance gains obtained from training word embeddings on the task corpora decreases as dataset size increases.

Having assessed the effect of differing architectural features between traditional methods and convolutional neural methods on text classification performance, I then evaluate techniques of data augmentation on the best performing models from these understandings. I explore existing data augmentation techniques for text classification, replicating the experiments of Wei and Zou (2019). My results generally confirm the findings in Wei and Zou (2019). However, the data augmentation methods proposed in Wei and Zou (2019) require external datasets and knowledge about the semantic properties of words. To this end, I propose a novel length-based data augmentation method in which a random word in a sentence is replaced with another word of similar length. This data augmentation technique is proposed because it does not require external linguistic knowledge about the semantic properties of words, and word length closely reflects the average information content and conceptual complexity of words in English (Piantadosi, Tily, and Gibson, 2011; Lewis and Frank, 2016). Consequently, word replacement with another word of similar length may generate a new training example that is more similar to the original example, compared to a training example generated by using a replacement word of vastly different length.

I evaluate this length-based technique on convolutional neural models. My results suggest that using word length as a heuristic for word replacement in generating augmented examples has a modest to significant effect on performance for the majority of evaluated classification tasks, with negative effects for only a single dataset.

Chapter 2

Background and Related Work

Many modern state-of-the-art approaches to natural language processing tasks use deep neural models. These methods often require large amounts of labeled data in order to train the models. However, data may not always be available in large quantities for a specific task (especially for languages other than English). Additionally, labeling data is often laborious and costly. Consequently, it is desirable to develop and understand techniques that allow these models to perform with comparable accuracy on significantly less data.

2.1 Text classification

One commonly studied task in natural language processing is text classification, also known as text categorization. This task involves assigning a category to a given text example, which can be important for applications such as spam detection.

I investigate the performance of models on sentiment classification and question classification tasks. Sentiment classification tasks involve categorizing examples according to whether the author's overall orientation toward an object is positive or negative (Jurafsky and Martin, 2009). Question classification tasks involve categorizing examples of questions according to the type of answer expected from the addressee (for example, whether the expected answer to a given question is a number or a description of some entity).

2.2 Feature extraction

2.2.1 Linguistic features

In the context of machine learning, a feature is a property of the input data. Features are used in machine learning to more easily capture the complexity of input data by decomposing the input into smaller, identifiable components. In natural language processing, features can include linguistic properties such as morphemes (the individual units that make up words), or larger aspects of a training example such as the presence or absence of specific words.

For text classification models, word vectors are used to encode words as the features of a document. Words are converted into numerical vectors that can be used to train the model.

2.2.2 One-hot representation

The simplest word vector representation is the one-hot vector encoding. The one-hot vector representation takes a vocabulary of size V. Each word in the vocabulary is assigned a numerical index. A word is then represented as a vector of size 1 by V, where the index corresponding to the given word is equal to 1 and all other indices are equal to 0.

2.2.3 Word embeddings

The one-hot vector representation is sparse and scales linearly with vocabulary size, making this approach undesirable for text corpora with large vocabularies. Consequently, word embeddings are another form of word vector representation used in natural language processing. Word embeddings project words from the one-hot vector representation into a lower dimensional vector space. These embeddings can be randomly initialized, or trained on a text corpus to capture semantic features of words. If word embeddings are trained, the embeddings for semantically similar words are often closer in Euclidean or cosine distance compared to words that are more semantically different.

One algorithm used to find word embedding representations for many NLP applications is *GloVe* (Pennington, Socher, and Manning, 2014), which produces word vectors from a text corpus by training on the corpus and learning the vocabulary vector representations from aggregated global word co-occurrence statistics. Pennington, Socher, and Manning (2014) also released sets of word embedding vectors for public use in training models. These sets of vectors are pretrained on large corpora such as Common Crawl web data.

2.3 Text classification models

In this research, I focus on traditional text classification methods such as the Naive Bayes classifier, and neural text classification methods such as convolutional and recurrent neural models.

In this section, the following notation is used. Given an input document d, a text classification model is trained to predict the class c within a set of possible classes C. The input document d consists of words $w_1, \ldots, w_N$, where each word w_i is represented by a word vector x_i. The word vectors $x_1, \ldots, x_N$ may be used directly as the set of input features F to the model, or additional processing may be used to convert $x_1, \ldots, x_N$ into a set of features $F = f_1, \ldots, f_T$.

2.3.1 Naive Bayes classifier

The Naive Bayes classifier is trained to predict the class c within the set of classes C that has the highest posterior probability given the document d, using Bayesian inference. The prediction $\hat{c}$ is given as

$$\hat{c} = \arg\max_{c \in C} P(c|d)$$

In calculating the probability $P(c|d)$, we use Bayes' rule, given below.

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Substituting the conditional probability equation for $\hat{c}$ into Bayes' rule, we obtain

$$\hat{c} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)} = \arg\max_{c \in C} P(d|c)P(c)$$

Intuitively, we choose the class c that maximizes the product of the prior probability of the class $P(c)$ and the likelihood $P(d|c)$ of the document being generated, given that d is of class c. To simplify the process of determining the set of features $F = f_1, \ldots, f_T$ for a given document, Naive Bayes classifiers use two assumptions:

1. *Co-occurrence assumption.* The Naive Bayes classifier is insensitive to feature co-occurrence: the probabilities of the features are independent given the class c. In other words,

$$P(f_1, f_2, \ldots, f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \cdots \cdot P(f_n|c)$$

2. *Word order.* The Naive Bayes classifier is insensitive to word order: a text document is represented as a "bag of words," or unordered set of words. Only the frequency of words, rather than the position of words in the document, is important for classification.

The class returned by a Naive Bayes classifier is therefore given by

$$c = \arg\max_{c \in C} P(c) \prod_{f \in F} P(f|c)$$

2.3.2 Neural models

Neural models are currently considered "state-of-the-art" (SOTA) models for most natural language processing tasks. Neural models first learn word vector representations, and learn to compose these word vectors to complete classification tasks (Kim, 2014).

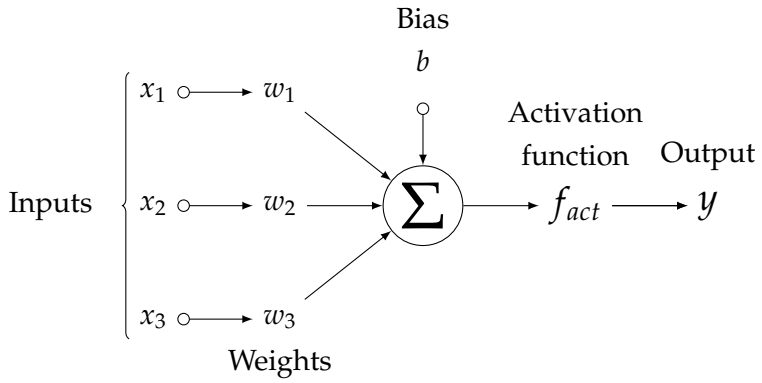

FIGURE 2.1: A single neural unit. Each input x_i is multiplied by its corresponding weight w_i. An activation function f_{act} is then applied to the weighted sum $\mathbf{w} \cdot \mathbf{x}$ plus a bias term b to produce the neuron output y. Figure derived from Reading (2019).

The smallest unit of a neural network is the *neuron*, which takes in an input and computes a weighted sum of the input data with a bias term. This weighted sum can be represented by the dot product:

$$z = \mathbf{w} \cdot \mathbf{x} + b$$

where $\mathbf{w}$ is a weight vector, b is a scalar bias term, and $\mathbf{x}$ is the vector of input data. For text classification models, the input data might be the set of features $F = f_1, \ldots, f_T$ extracted from the word vectors $x_1, \ldots, x_N$ for a given document d. The neuron then computes an output value by applying a nonlinear *activation function* (f_{act}) to z, resulting in an *activation value* (a), where $a = f_{act}(z)$. The final output of the single neuron is then:

$$y = a = f_{act}(z)$$

(Jurafsky and Martin, 2009). The basic neural network is the *feed-forward neural network*, which combines multiple neurons into a network. The inputs of neurons within the network can be inputs to the entire network and/or outputs of other neurons. These neurons are organized into *layers* of neurons, where a layer is a set of neurons that are not connected to each other.

The feed-forward neural model typically consists of an input layer, followed by some number of hidden layers, followed by an output layer. A single layer is only capable of learning a linear function to fit the training data. In order to learn more complex functions, neural models typically incorporate one or more *hidden layers* into the model architecture. Multi-layer neural models use a nonlinear activation function to introduce nonlinearity and additional complexity.

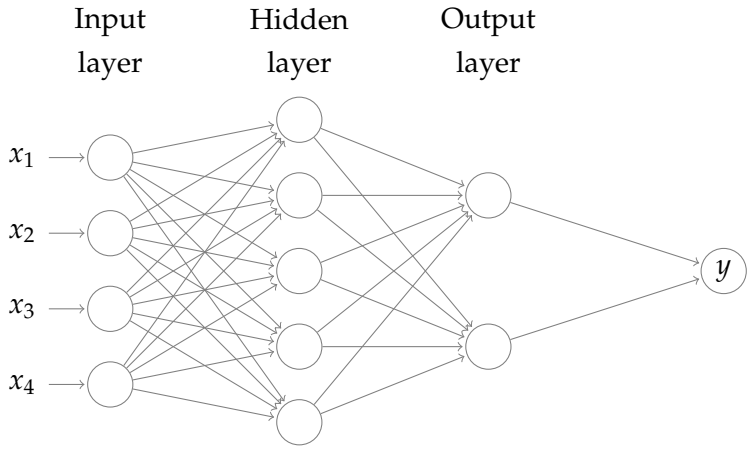

FIGURE 2.2: A basic feed-forward neural network, where each arrow represents multiplication by a weight, addition of a bias term, and application of a nonlinear function. Figure derived from Fauske (2006).

A neural text classification model is typically trained on a provided text corpus to complete a specific classification task. During training, the neural model learns the weights of the neurons from the training data. A predefined *loss function* is used as a metric of the model's performance on the training data to model the difference between the neural model's output and the accepted "correct" output. Various methods are then used for *error backpropagation*, in which the model's weights are adjusted to minimize the error obtained by the model on the training dataset.

Convolutional neural networks

The standard feedforward neural network model can only consider a fixed-length input. It cannot model variable-length context such as a sentence, which is important for modeling natural language.

Convolutional neural models (CNNs) have previously demonstrated good performance on NLP tasks such as text classification, outperforming the state of the art on several sentiment analysis and question classification tasks (Kim, 2014).

CNNs operate by applying filters to the input word vectors $x_1, \ldots, x_N$ using a *sliding window*. The network slides, or convolves, each filter across the input, taking the dot product over the filter and the input word vectors encompassed by the sliding window at a given position. Using these filters, the CNN reduces the input document into a smaller set of features important for classification, where each filter is associated with a specific feature of the document. The network is then trained to activate the relevant filters for the features of a given input document (Li, Krishna, and Xu, 2020).

CNNs also often apply pooling layers to the result of the convolution, which are used to yield an output of fixed size from inputs of variable size (a fixed dimension output is often

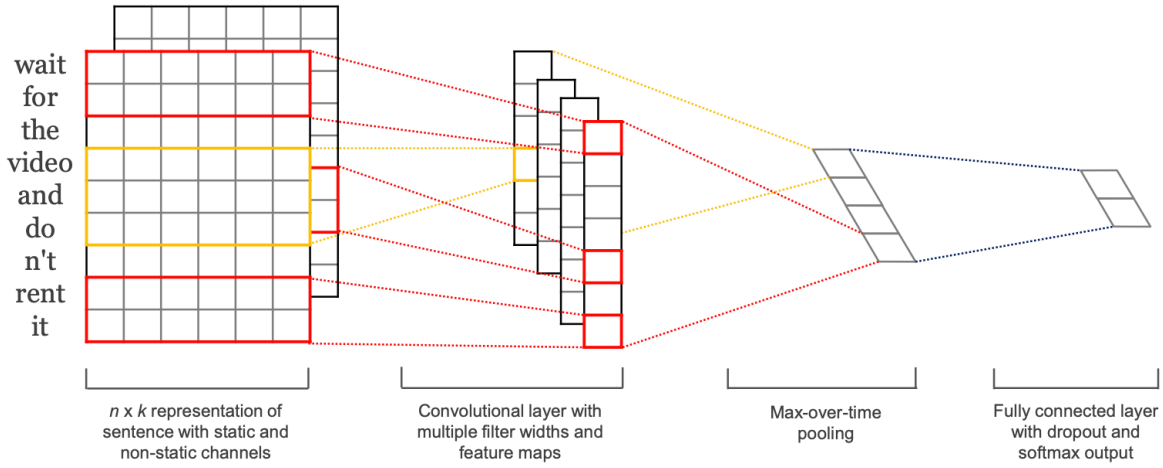

FIGURE 2.3: A convolutional neural network for text classification. Figure from Kim (2014).

required for classification). These pooling layers are also used to further reduce the dimensionality of the output while preserving relevant information. The most common form of pooling is max pooling, where the maximum is taken over the result of the convolutional layer.

CNNs are commonly used for image classification, where the filters are used to detect features such as edges or patterns. When used in natural language processing, CNNs typically use word vector representations such as pretrained word embeddings, and the filters are used to detect semantic or syntactic linguistic features.

Recurrent neural networks

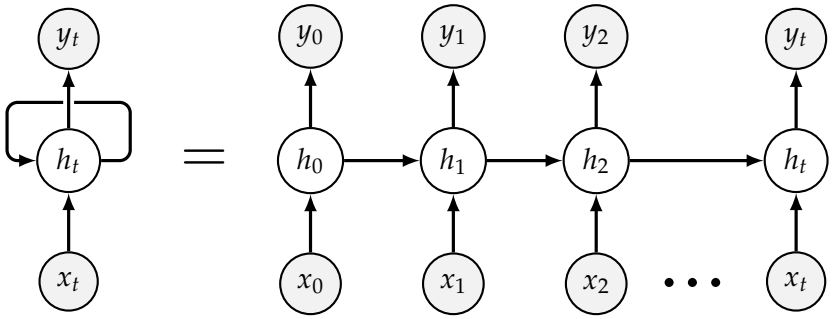

FIGURE 2.4: A basic recurrent neural network. At each timestep t, the activation value y_t depends on the current input x_t and the hidden state of the previous timestep h_{t-1}. Figure derived from Olah (2015).

Recurrent neural models are a variant of neural network models that incorporate the sequential nature of human language processing. At any given point in time t, the neural network maintains a representation of the input $x_1, \ldots, x_t$ that has been processed so far in a *hidden state* h_t, or

the *context* for the text processed thus far. This hidden state serves as a form of memory for the neural network and is fed in as input at each additional timestep in addition to the new input x_{t+1}, hence the term *recurrence*. The hidden state in the RNN enables the model to incorporate linguistic information extending arbitrary distances (Jurafsky and Martin, 2009).

The long short-term memory (LSTM) recurrent neural network is a variant of the standard vanilla RNN model. LSTMs improve on the standard RNN by using a series of "gates" that determine the extent to which the network should preserve specific textual information across timesteps. These gates enable the network to "forget" unnecessary information from the context, and add new information that may be necessary for future decisions (Levy, 2020; Jurafsky and Martin, 2009).

2.4 Data augmentation

Data augmentation is a technique in which additional training data is created by manipulating existing training data, either by making small modifications to the original examples, or by using the existing data as a template. It has been widely used for image classification (Ciresan et al., 2010; Krizhevsky, Sutskever, and Hinton, 2017; Zhang et al., 2017; Summers and Dinneen, 2018) and natural language processing (Lu et al., 2006), particularly low-resource domains (Fadaee, Bisazza, and Monz, 2017).

2.4.1 Data augmentation in natural language processing

Previous work on data augmentation in natural language processing explores a variety of techniques to manipulate parts of training examples to generate new examples. Andreas (2019) uses an algorithm to identify sentence fragments that appear in similar environments, proposing that swapping such fragments with other fragments that appear in similar environments should create similar and plausible examples. Xie et al. (2017) use data noising as smoothing as a form of data augmentation. Synonym replacement as a method to generate new training examples is also a widely explored technique (Kolomiyets, Bethard, and Moens, 2011; Zhang, Zhao, and LeCun, 2015; Wang and Yang, 2015). To find synonyms, Kolomiyets, Bethard, and Moens (2011) use WordNet (Miller, 1995), which organizes words by semantic distance; Zhang, Zhao, and LeCun (2015) use a thesaurus derived from WordNet. Wang and Yang (2015) use a k-nearest-neighbors approach with word embeddings to determine replacement synonyms.

Other related studies use neural models to generate augmented data. Kobayashi (2018) proposes and investigates the technique of contextual data augmentation, which replaces words with words predicted from a bidirectional language model, according to the context of the original word. Yu et al. (2018) implement backtranslation, which uses two translation models to generate paraphrased examples: one to translate the original example from English to another language, and one to translate back from that language to English. Kolomiyets, Bethard, and

Moens (2011) also explore a form of synonym replacement using an external language model and find that intersecting the results of the language model with WordNet can lead to more stable results for some tasks.

2.4.2 Easy Data Augmentation

Wei and Zou (2019) investigate four simple text editing techniques for data augmentation, collectively titled Easy Data Augmentation (EDA). The authors evaluate the performance of a long short-term memory recurrent neural network and a convolutional neural network on five text classification tasks. Easy Data Augmentation randomly applies one of the following operations to each sentence in the training dataset to generate additional training examples:

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.

Wei and Zou (2019) apply the "standard" form of Easy Data Augmentation, which randomly chooses between the four techniques to generate augmentations for a training example, to various increments of the training datasets. The authors also evaluate the effect of applying each Easy Data Augmentation method individually to the datasets. Other experiments in Wei and Zou (2019) vary the number of augmentations per sentence and the number of additional sentences generated per training example to determine the values for these parameters that yield the highest test accuracy. Without augmentation, the evaluated RNN and CNN models achieve an average accuracy of 88.3% using 100% of the training data, while models trained on EDA-augmented data achieve an average accuracy of 88.6% while using only 50% of the training data. Overall, Easy Data Augmentation leads to an average improvement in accuracy of 0.8% for complete datasets, and an average accuracy improvement of 3.0% for "small" fractions of the training datasets containing 500 training examples.

All of the Easy Data Augmentation methods (synonym replacement, random insertion, random swap, and random deletion) are individually capable of increasing the accuracy of both CNNs and RNNs for the given classification tasks. From the perspective of human language learning, it seems unlikely that all techniques would yield accurate examples for text classification tasks, since methods such as random insertion can potentially distort the labels and

introduce noise into the training corpus. However, ablation studies conducted by Wei and Zou (2019) show that Easy Data Augmentation generally tends to conserve class labels for augmented sentences. These results suggest that Easy Data Augmentation acts as a regularizer and improves the model's generalization abilities.

2.5 Analyzing Easy Data Augmentation and model features

The methods presented in Wei and Zou (2019) and Andreas (2019) are "easy" techniques of data augmentation informed by linguistic knowledge. Such simple methods of data augmentation are of particular interest because they do not require additional language models (which are resource intensive to train) in order to generate new training examples.

Much of the existing literature for applying data augmentation to text classification tasks focuses on seeking additional methods of data augmentation to improve model performance. However, it remains unclear as to which architectural features of classification models are most important to obtain the high performance achieved by the current state-of-the-art models. For the specific text classification tasks in Wei and Zou (2019), there are no comprehensive fine-grained evaluations of the various factors contributing to the performance of text classification models. The most similar work that conducts a comparison of traditional and deep learning methods for text classification tasks is Zhang, Zhao, and LeCun (2015), which compares character-level CNNs to other text classification models such as word-level CNNs and multinomial logistic regression classifiers, for several sentiment and topical text classification datasets. However, this study uses several large-scale datasets ranging in size from $120,000$ to $3,600,000$ training samples. These datasets are much larger in magnitude compared to the datasets used in Wei and Zou (2019), so it is difficult to assess how their results might compare for models trained on smaller datasets.

Given currently existing research, it is of interest to investigate what factors contribute to text classification models' performance, as well as explore other methods of "easy" data augmentation (informed by linguistic knowledge) that may increase the accuracy of text classification models.

Chapter 3

Models and Methods

3.1 Models

For these experiments, I examine both traditional and neural models used for text classification: Naive Bayes classifiers, convolutional neural networks (CNNs), and long short-term memory recurrent neural networks. Unless indicated otherwise, I use the term RNN to refer to the long short-term memory recurrent neural network.

3.1.1 Naive Bayes

The model used as a baseline in the Naive Bayes experiments is a unigram, multinomial Naive Bayes (NB) classifier using raw word counts as features. This baseline version of Naive Bayes uses only the frequencies of words in the training examples to predict classification for a document, does not take word order into account, and uses a one-hot vector representation of the vocabulary. The frequency of words is taken into account since a word reflecting the overall classification of the example is likely to be used more frequently than the opposing term. Unigrams are used as features in order to examine the performance of the NB classifier on a corpus without regard to word order.

I also determine the performance of the multinomial Naive Bayes classifiers using two types of word pairs as features:

1. **Bigrams formed from consecutive word pairs.** This determines the effect of preserving some word order within each example.

2. **All possible word pair combinations across each example.** This determines whether word pairs within the sentence are important features for the classification tasks.

3.1.2 Neural models

For consistent comparison of results, I implement the CNN and RNN architectures used in Wei and Zou (2019), reproduced here for convenience.

- **CNN architecture.** The CNN architecture used in these experiments consists of an 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 architecture.** The RNN architecture used in these experiments consists of an 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 crossentropy loss function with the adam optimizer. We use early stopping with a patience of 3 epochs.

3.2 Feature extraction

For models using pretrained word embeddings, I use the set of pretrained GloVe word vectors from Pennington, Socher, and Manning (2014) trained on 840 billion tokens of cased Common Crawl web data, with a vocabulary size of 2.2 million words. Unless specified, these word embeddings are not further trained on the task corpora.

3.2.1 Handling OOV tokens

If a word in a test corpus is not present in the original training corpus, the text classification model is unable to recognize the word and the word is therefore considered *out-of-vocabulary* (OOV). Unknown words are mapped to a designated OOV token.

3.2.2 Word frequency threshold

Some words in the training corpus may have low frequency in comparison to other words. These low-frequency words can be of lower importance for the text classification task, compared to more common words. For example, for a sentiment classification task, the word *bad* is generally more helpful to determine the sentiment of the example and would likely appear more frequently in a text corpus, compared to the utility and frequency of word *marmot*. To take the frequency of words into account, some studies apply a threshold value on the frequency of a word in the training corpus to determine whether a token should be mapped to the OOV token. However, Wei and Zou (2019) do not use a threshold on word frequency to determine OOV tokens. Rather, a word is considered OOV if it is not in the GloVe dictionary, and is instead mapped to a zero vector word embedding. Consistent with the approach used in Wei and Zou (2019), I also do not use a frequency threshold to determine OOV tokens, and map each token to a vocabulary index without regard to frequency in the corpus.

3.2.3 Stop words

Stop words are frequent words such as *the* or *a*. Consistent with Wei and Zou (2019), I ignore stop words when choosing words within a sentence for data augmentation purposes. The stop words used in experiments are listed in Appendix A.

Chapter 4

Experiments

4.1 Datasets

I use the following five text classification tasks, consistent with the tasks used in the experiments performed by Wei and Zou (2019).

- **SST-2.** The Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013a) is a classification dataset consisting of movie reviews and their sentiment classification (positive or negative). I use the two-class version of the SST dataset (SST-2) in which each review is classified as either strictly positive or strictly negative. Examples from the dataset include the review *a masterpiece four years in the making*, which is classified as *positive*, and *though everything might be literate and smart it never took off and always seemed static*, which is classified as *negative*.

- **CR.** The Customer Review (CR) dataset (Hu and Liu, 2004; Liu et al., 2015) is a binary classification dataset. This dataset consists of customer reviews collected for nine products, such as a Canon camera, Apple iPod and an Internet router. The objective is to classify a given review as positive or negative. Examples from the dataset include the review *i really love this phone*, which would be classified as *positive*, and the review *headphones as with most headphones that come with any type of music player i recommend throwing these into the nearest river unless you like having hard round pieces of plastic in your ears*, which would be classified as *negative*.

- **SUBJ.** The Subjectivity-Objectivity (SUBJ) dataset (Pang and Lee, 2004) is a binary classification dataset. The objective of the classification task is to correctly distinguish between subjective and objective statements about movies. The subjective statements are movie review *snippets* collected from the website www.rottentomatoes.com, while the (mostly) objective statements are sentences from plot summaries collected from the Internet Movie Database (IMDB) website www.imdb.com.

- **TREC.** The Text REtrieval Conference (TREC) dataset (Li and Roth, 2002) is a question classification dataset, consisting of questions and possible classifications for the type of

question being asked. Two forms of TREC exist (TREC-6 and TREC-50), where the number indicates the number of possible classifications for the type of question in each example. I use TREC-6 for my experiments, with the following possible classifications:

  - ABBR - abbreviation
  - DESC - description and abstract concepts
  - ENTY - entities
  - HUM - human beings
  - LOC - locations
  - NUM - numeric values

  For example, the question *where is the highest point in japan* is classified as *LOC* because the answer would be a location, while the question *what is a transistor* is classified as *DESC* because the answer would be a description of an entity. In this thesis, all uses of the term TREC refer to the TREC-6 dataset. Within each of the training and test dataset, the proportions of each question type are not equal. However, the proportions of each question category remain constant between the training and test sets.

- **PC.** The Pro-Con (PC) dataset (Ganapathibhotla and Liu, 2008) is a binary classification dataset consisting of customer reviews of services and products. The objective of this classification task is to classify whether the given example is a pro or a con for the product or service being reviewed. Examples from the dataset include the review *solid easy to use phone with speakerphone feature*, which would be classified as *pro*, and the review *worst user interface of any phone I have ever used*, which would be classified as *con*.

Dataset	$\lvert V_{train} \rvert$	Class	# Training Examples	# Test Examples
SST-2	13,898	Positive	3,863	876
		Negative	3,530	873
		Total	7,393	1,749
CR	3,761	Positive	1,205	130
		Negative	658	78
		Total	1,863	208
SUBJ	20,284	Subjective	4,500	500
		Objective	4,500	500
		Total	9,000	1,000
TREC	8,173	ABBR	86	9
		DESC	1,162	138
		ENTY	1,251	94
		HUM	1,223	65
		LOC	835	81
		NUM	896	113
		Total	5,453	500
PC	8,193	Pro	20,174	2,257
		Con	19,249	2,246
		Total	39,423	4,503

TABLE 4.1: Magnitude of the text classification datasets in vocabulary size and in number of training and test examples for each possible class. $\lvert V_{train} \rvert$ is the size of the training dataset vocabulary.

4.2 Baselines

4.2.1 Naive Bayes classifier

This group of experiments studies whether it is possible to to use a less complex model requiring less computational power to achieve comparable performance to neural models on the classification tasks. In other words, these experiments establish a baseline to determine the relative "difficulty" of the text classification tasks. Unigram, bigram, and random word pair variants of the multinomial Naive Bayes classifier are applied to various subsets of the task corpora.

4.2.2 Removing Naive Bayes classifier assumptions

Naive Bayes classifiers make two major assumptions about the training corpora:

1. *Co-occurrence assumption: The features within a particular text example (such as unigrams or bigrams) are independent, given that we know the classification for that example.*

   To study the effect of the co-occurrence assumption, I compare the accuracies of 1) the unigram Naive Bayes classifier and 2) a CNN trained with one-hot vector representations, with random word order for each example. The CNN removes the assumption that the features are independent given the classification for that example by "learning" how features within a text example combine to output a final classification. Using random word order prevents the CNN model from taking word order within examples into account, isolating the effect of the co-occurrence assumption on task accuracy.

2. *Word order assumption. The order of words within the corpus does not matter.*

   To study the effect of the word order assumption, I investigate the effect of word order as a training feature. I compare the accuracies of 1) the CNN trained with one-hot vector representations, with random word order for each example and 2) a CNN trained with one-hot vector representations, with the original word order for each example.

Subsequent experiments preserve original word order unless otherwise specified.

4.2.3 Word embeddings

These experiments are used to determine the effect of the pretrained GloVe embeddings used in neural models. The pretrained GloVe embeddings differ from the traditional one-hot vocabulary representations in two ways.

1. *Lower dimensional representation: Word embeddings project vocabulary word vectors to a lower dimensional space than the one-hot vector representation.*

   To study the effect of a lower dimensional word representation space, I compare the accuracies of 1) the CNN using one-hot vector representations of the vocabulary, where the dimensions of the vector representations are on the order of training vocabulary size, and 2) a CNN using lower dimension (300d) word embeddings with randomly generated weight values. By using randomly generated embedding weights, this experiment isolates the effect of projecting vocabulary representations into a lower dimensional space.

2. *Latent linguistic information: Pretrained GloVe word embeddings are pretrained on separate corpora, allowing the embeddings to encode predefined linguistic information such as the semantic relationships between words.*

To study the effect of the pretrained aspect of GloVe embeddings, I compare the accuracies of 1) the CNN using randomly generated 300d word embeddings and 2) a CNN using pretrained 300d GloVe embeddings. This experiment isolates the pretrained aspects of the GloVe embeddings by preserving the embedding dimensions between the two models.

4.3 Augmentations

4.3.1 Easy Data Augmentation

In this group of experiments, I replicate the experiments in Wei and Zou (2019). I examine the four methods of "easy" data augmentation: synonym replacement, random deletion, random insertion, and random swap, under various dataset and parameter settings.

4.3.2 Length-based data augmentation

I propose and examine a novel length-based method of "easy" data augmentation. To generate a new training example, this technique randomly chooses n words from the sentence that are not considered stop words, and replaces each of these words with a new word chosen at random from a set of words that are of similar length. This data augmentation technique is proposed because it does not require external knowledge about the semantic properties of words. Additionally, word length closely reflects the average information content and conceptual complexity of words in English (Piantadosi, Tily, and Gibson, 2011; Lewis and Frank, 2016). Consequently, word replacement with another word of similar length may generate a new training example that is more similar to the original example, compared to a training example generated by using a replacement word of vastly different length.

I train and evaluate CNN models [1] with the architecture used in Wei and Zou (2019) on these augmented datasets. I use two different sets of replacement words to generate new training examples:

1. *Words from the training dataset.* For each training dataset, I generate a dictionary of all words of length within two characters of the longest word in the given training corpora. To generate new training examples, one or more randomly selected words in the original training example are replaced with a randomly selected word from the training dataset vocabulary with a length within a certain number of characters of the original word.

   I perform three experimental variations for this set of replacement words, using replacement words with lengths within $0, 1$, or 2 characters of the original word's length.

[1] Due to time constraints on experimentation, only CNN models were evaluated for this method of data augmentation. Future work could potentially include an investigation into how this method would affect the performance of other models such as RNNs.

2. *Words from the GloVe dictionary.* I generate a dictionary of all words in the GloVe dictionary of length within two characters of the longest word in the training corpora for the text classification tasks. To generate new training examples, one or more randomly selected words in the original training example are replaced with a randomly selected word of the same length from the GloVe dictionary.

Chapter 5

Results and Discussion

In this section, *Dataset size* indicates the size of training dataset. The value N_{train} is the number of examples in the training dataset.

5.1 Naive Bayes classifier

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	71.2	66.2	82.7	39.0	80.3	67.9
2000	73.6	74.6	86.6	47.8	85.0	73.5
5000	73.6	79.5	88.5	56.0	87.1	76.9
Full set	73.6	81.7	89.7	56.4	89.6	78.2

TABLE 5.1: Accuracies (%) of the unigram multinomial Naive Bayes classifier on the five classification tasks, for different sizes of the training datasets.

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	63.9	52.9	54.7	20.2	71.7	52.7
2000	72.1	56.3	62.9	24.0	68.0	56.7
5000	72.1	59.2	70.5	27.8	73.6	60.7
Full set	72.1	61.2	77.2	29.2	82.6	64.5

TABLE 5.2: Accuracies (%) of the bigram multinomial Naive Bayes classifier on the five classification tasks, for different sizes of the training datasets.

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	62.5	51.9	51.8	19.6	60.7	49.3
2000	68.8	51.9	54.8	20.8	58.1	50.9
5000	68.8	55.8	57.9	23.8	67.4	54.7
Full set	66.8	56.3	64.2	22.0	78.9	57.6

TABLE 5.3: Accuracies (%) of the random word pairs multinomial Naive Bayes classifier on the five classification tasks, for different sizes of the training datasets.

Accuracies of the Naive Bayes classifiers are shown in in Table 5.1, Table 5.2, and Table 5.3. A comparison of the three variations on the Naive Bayes classifier for each dataset size is shown in Figure 5.1. Of the three Naive Bayes variations, the unigram multinomial Naive Bayes classifier yielded the highest accuracies for all training dataset sizes across the classification tasks. The results suggest that using individual words as features, rather than the co-occurrence of words, is a better predictor of the class for a given example when using the Naive Bayes classifier.

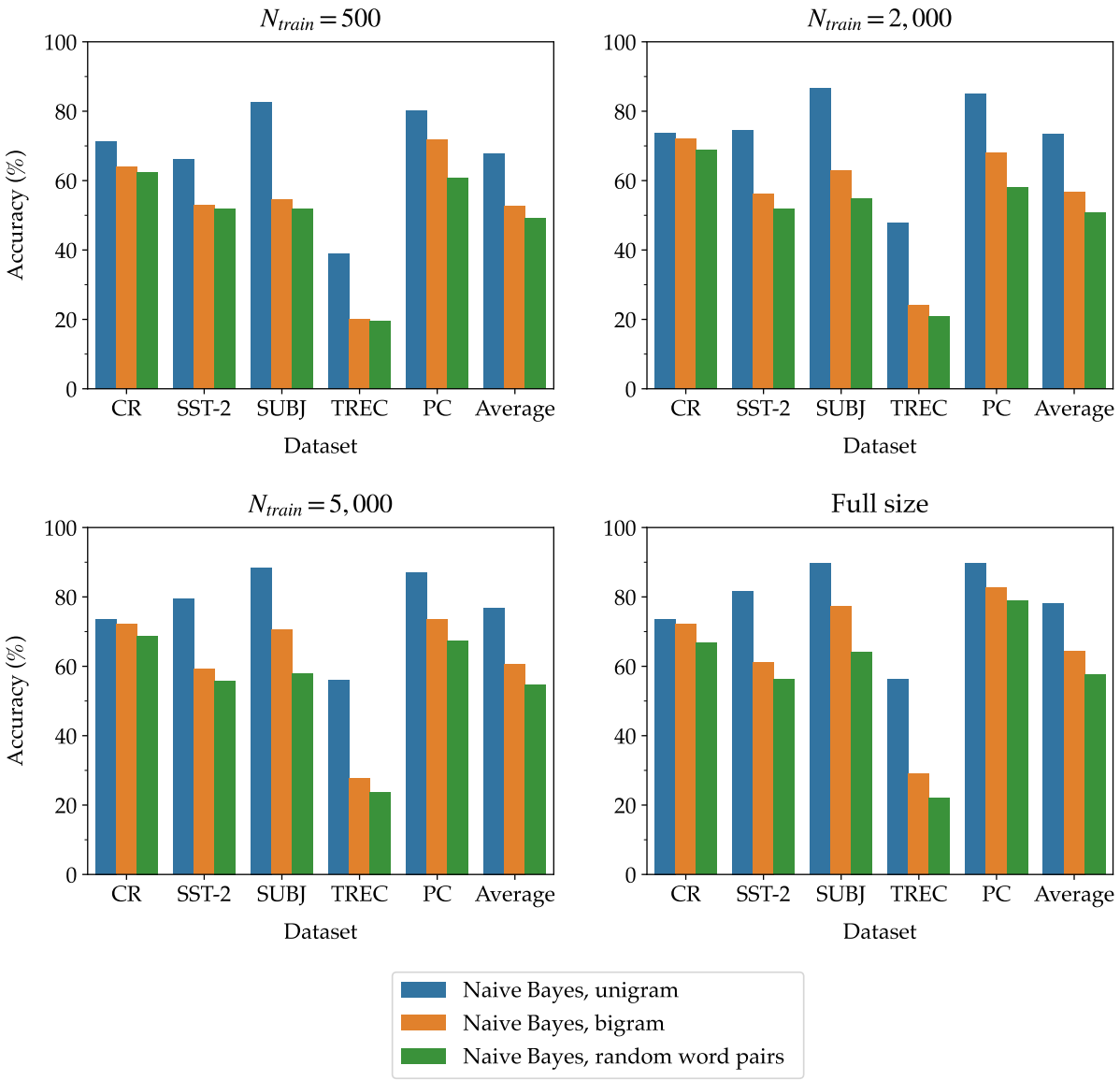

FIGURE 5.1: Average performances (%) of the unigram, bigram, and random word pairs Naive Bayes classifiers across the five classification tasks, for different sizes of the training datasets.

5.2 Removing Naive Bayes classifier assumptions

5.2.1 Co-occurrence assumption

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	50.4	60.0	51.6	22.4	51.8	47.2
2000	50.5	61.6	55.3	22.5	55.0	49.0
5000	50.1	62.5	73.5	24.5	65.2	55.2
Full set	50.2	62.7	72.5	26.4	84.7	59.3

TABLE 5.4: Accuracies (%) of the one-hot CNN model trained with random word order on the five classification tasks, for different sizes of the training datasets.

In this experiment, I investigate the effect of the co-occurrence assumption on the accuracy of text classification models. I compare the accuracies obtained by 1) a multinomial unigram Naive Bayes classifier and 2) a CNN model using one-hot vector representations of the vocabulary and random word order for each training example. Accuracies of the models in this experiment are shown in Table 5.1 and Table 5.4, with comparisons for the models summarized in Figure 5.2. Across all classification tasks and dataset sizes, the results suggest that simply replacing the unigram Naive Bayes classifier architecture with a CNN architecture decreases performance significantly. Interestingly, models trained on the smallest size of the SST-2 dataset and the full PC dataset have the smallest decrease in accuracy compared to the changes for other combinations of classification tasks and dataset sizes.

5.2.2 Word order assumption

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	50.6	60.9	50.3	23.1	52.9	47.6
2000	50.9	62.6	50.0	23.5	54.1	48.2
5000	50.6	62.7	72.4	21.5	65.1	54.5
Full set	50.2	62.5	73.5	30.5	86.2	60.6

TABLE 5.5: Accuracies (%) of the one-hot CNN model trained with original word order on the five classification tasks, for different sizes of the training datasets.

In this experiment, I investigate the effect of the word order assumption in the Naive Bayes classifier model. To do so, I compare the task accuracies of 1) a CNN model using a one-hot

vector representation of the vocabulary with random word order for each example, and 2) a CNN model using a one-hot vector representation of the vocabulary trained with the original word order for each example. The accuracies of the two models are shown in Table 5.4 and Table 5.5, and comparisons of the models for each dataset size are displayed in Figure 5.2. Averaging across the classification tasks, the results suggest that using word order as a feature does not appear to have a large effect on the accuracy. The presence of word order as a feature appears to have a different effect on the accuracies achieved for individual classification tasks. For example, for smaller dataset sizes of the SUBJ dataset, the accuracy of the CNN model trained on data with random word order is higher compared to the CNN model trained on data where original word order is preserved. In comparison, for the full size TREC dataset, the CNN trained on ordered data shows a 4.1% increase in accuracy compared to the CNN trained on randomly ordered data. One caveat is that both one-hot CNN models perform close to chance, making it difficult to generalize the conclusions here to other settings.

For subsequent experiments, all models are trained on the task corpora with original word order in each example unless stated otherwise.

5.3 Word embeddings

5.3.1 Effect of lower dimension vector representations

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	54.9	62.5	67.1	66.7	72.9	64.8
2000	64.8	63.4	79.0	79.9	79.8	73.4
5000	68.9	64.3	82.5	85.2	84.3	77.0
Full set	70.7	63.3	84.5	86.8	89.3	78.9

TABLE 5.6: Accuracies (%) of the random word embedding CNN model on the five classification tasks, for different sizes of the training datasets.

In this experiment, I determine the effect of using lower dimension vector representations of the vocabulary on model accuracy for the text classification tasks. I compare the accuracies of 1) a one-hot CNN model and 2) a CNN model with randomly generated word embeddings of lower dimension (300 dimensions). Weights in the embedding layer are not changed during training. The accuracies of the two models are shown in Table 5.5 and Table 5.6, and comparisons of the models for each dataset size are summarized in Figure 5.2. The results show that using lower-dimensional word embedding representations has a large positive effect on model accuracy for the text classification tasks. Averaging across the classification tasks, the

models with random word embeddings show a significant increase in accuracy compared to the one-hot models, with the increase in accuracies ranging from 17.2% to 25.2%. However, the difference in accuracies varies among classification tasks. Using a lower dimensional word embedding representation has a larger positive effect on model performance for the TREC task compared to the other tasks. For example, the differences in accuracies for the SST-2 task range from 0.8% to 1.6%, while the differences in accuracies for the TREC task are larger and range from 43.6% to 63.7%. Additionally, within tasks, the effect of using lower dimensional word embeddings changes as dataset size increases. For the CR task, the increase in accuracy obtained by using random word embeddings increases as dataset size increases, while for the PC task, the increase in accuracy from using random word embeddings decreases significantly for the full dataset size compared to other dataset sizes.

The greater performance gains observed for the TREC question classification task may occur because question classification may rely on the presence of a word from a well-defined set of question words (such as *who, why, where*) to determine the category for an example. Representing words with a lower number of dimensions would allow the model to better learn the features of these keywords during training. In contrast, sentiment classification focuses on the positive or negative meaning of words, which may be less easily captured by only using fixed lower dimension vector representations.

5.3.2 Effect of using pretrained embeddings

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	72.0	65.1	82.9	67.6	83.0	74.1
2000	79.7	76.1	87.9	85.9	87.1	83.3
5000	82.8	77.7	89.5	90.0	89.5	85.9
Full set	83.1	77.1	90.3	90.8	92.2	86.7

TABLE 5.7: Accuracies (%) of the untrained GloVe embedding CNN model on the five classification tasks, for different sizes of the training datasets.

In this experiment, I determine the effect of using pretrained GloVe word embeddings that encode linguistic knowledge into the word representations. I compare the accuracies of 1) a CNN with randomly generated word embeddings of the same dimension as the GloVe embeddings (300 dimensions) and 2) a CNN with pretrained GloVe embeddings. Weights in the embedding layer are not changed during training. The accuracies of the models for this experiment are shown in Table 5.6 and Table 5.7, and model comparisons for each dataset size are summarized in Figure 5.2. Averaging across the tasks, the results demonstrate that using pretrained word embeddings with encoded linguistic knowledge increases accuracy compared to using

randomly initialized embeddings. Out of the four possible dataset sizes, the highest gains in average accuracy are obtained with the $N_{train} = 2,000$ dataset size.

We see that the accuracies obtained in the CR and SST-2 tasks are the main contributors to the peak in accuracy seen with the $N_{train} = 2,000$ dataset size. Within CR, this dataset size has a higher increase in accuracy from using GloVe embeddings, compared to increases in accuracies for the other sizes. The $N_{train} = 5,000$) and full dataset sizes have approximately the same increase in accuracy, and larger increases in accuracy compared to the $N_{train} = 500$ dataset size. Within SST-2, the change in accuracy increases from 2.2% for the $N_{train} = 500$ dataset size to 14.4% for the $N_{train} = 2,000$ dataset size. However, for larger increments of the SST-2 dataset, the change in accuracy decreases as dataset size increases. For the remaining classification tasks (SUBJ, TREC, and PC), the increase in accuracy obtained from using GloVe embeddings decreases as dataset size increases.

Comparing the change in accuracy across tasks, the CR and SST-2 tasks tend to obtain larger increases in accuracy compared to SUBJ, TREC, and PC. Interestingly, SUBJ achieves modest accuracy gains from using GloVe embeddings, with the exception of the $N_{train} = 500$ dataset size, where it has the second largest increase in accuracy out of the five classification tasks. Additionally, the results of the two word embedding experiments demonstrate that for different classification tasks, different effects are observed for changing the dimensionality and linguistic knowledge encoded in word embeddings. For the TREC task, CNN models obtain a much larger increase in accuracy from using lower dimensional word embeddings, compared to the increase in accuracy from using linguistic knowledge in pretrained word embeddings. In comparison, the CR task appears to benefit equally from both lower dimensionality and the encoded linguistic knowledge.

The greater performance gains observed for sentiment classification compared to question classification may occur because sentiment classification depends on the positive or negative connotations, and therefore semantic meaning, of the words in a given example. The semantic meaning encoded in pretrained word embeddings would therefore contribute significantly to the model's ability to predict sentiment. Such linguistic knowledge in pretrained word embeddings may have less of an effect for question-type tasks, where the meaning of words plays a smaller role in classification.

5.3.3 Effect of training word embeddings

	Dataset					
Dataset size	CR	SST-2	SUBJ	TREC	PC	*Average*
500	72.1	66.1	84.1	71.7	84.2	75.6
2000	80.4	78.1	88.6	86.6	88.1	84.4
5000	84.3	76.5	90.0	90.0	90.2	86.2
Full set	84.8	77.8	91.5	90.0	92.8	87.4

TABLE 5.8: Accuracies (%) of the GloVe embeddings CNN model with embedding layer trained on the task vocabulary on the five classification tasks, for different sizes of the training datasets.

This experiment seeks to determine the effect of training word embeddings while training the model on the task corpora. In this experiment, I compare the accuracies of 1) a CNN model using GloVe embeddings with the weights in the embedding layer kept constant during training, and 2) a CNN model using GloVe embeddings with the weights in the embedding layer further trained on the task corpora. The accuracies of models used in this experiment are shown in Table 5.7 and Table 5.8, with comparisons of the models for each dataset size summarized in Figure 5.2. Averaging across the classification tasks, training embeddings on the task corpora has a generally positive effect on accuracy, with the greatest positive effect on accuracy for smaller sized datasets. For individual tasks, the effect of training word embeddings on the task corpora varies. For example, for the CR dataset, as the size of the datasets increases, the difference in accuracy obtained from training word embeddings increases, from $+0.1\%$ to $+1.7\%$. This increase likely occurs because the embedding weights become better tailored to the task with the greater availability of data in larger subsets of the training datasets. In comparison, for TREC and PC, the change in accuracy obtained from training word embeddings on the task corpora decreases as dataset size increases.

5.4 Summary of model comparisons

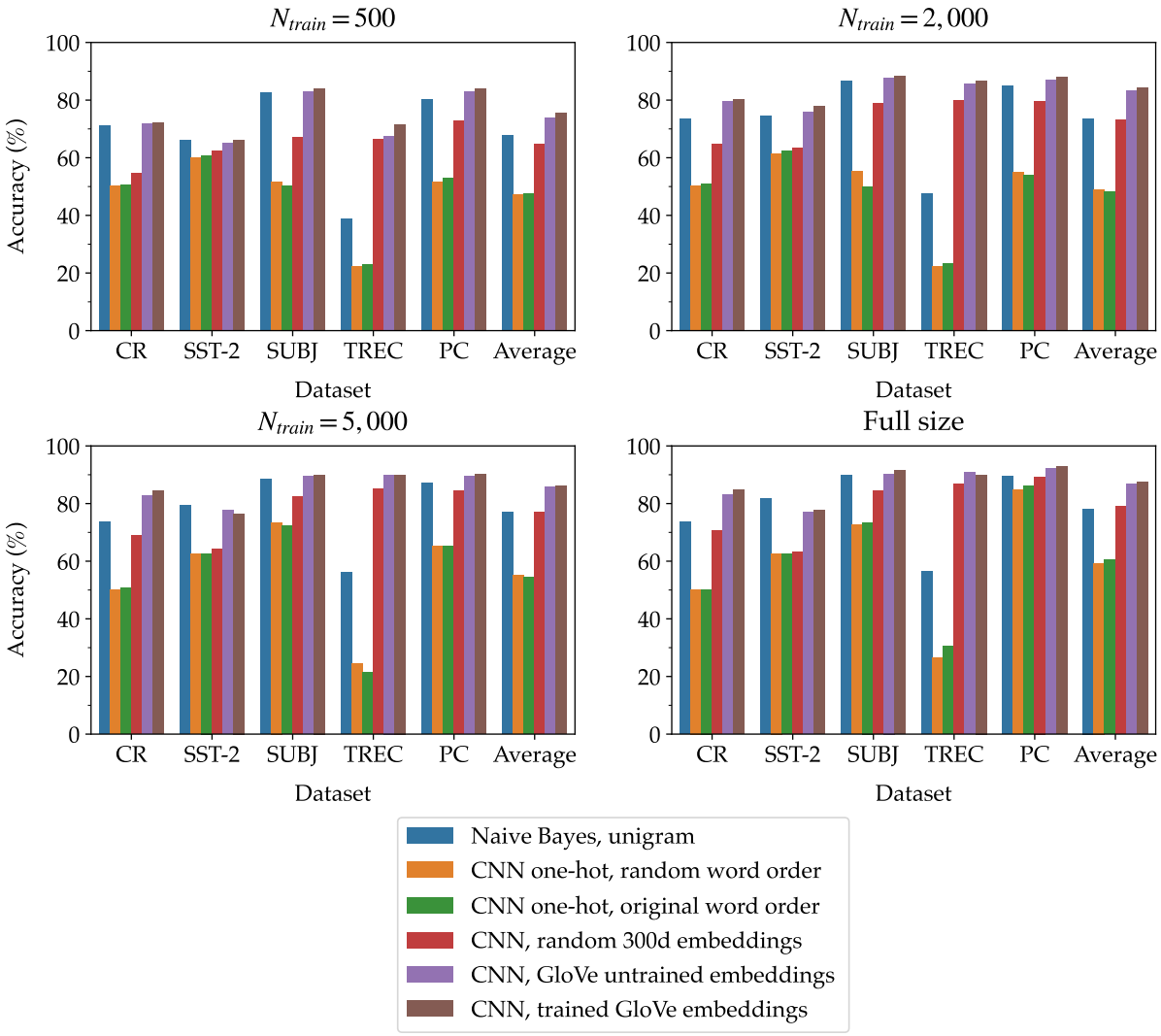

FIGURE 5.2: Average performances (%) of text classification models under various settings across the five classification tasks, for different sizes of the training datasets.

The results of the text classification models presented in this section are summarized in Figure 5.2.

5.5 Easy Data Augmentation

5.5.1 Standard CNN model with Easy Data Augmentation methods

In these experiments, I replicate the Easy Data Augmentation (EDA) experiments presented in Wei and Zou (2019). In this section, the term *standard EDA* refers to data augmentation that randomly selects from one of four methods (synonym replacement, random insertion, random deletion, and random swap) to generate additional training examples. Standard EDA was performed on these partitions to produce an augmented training dataset.

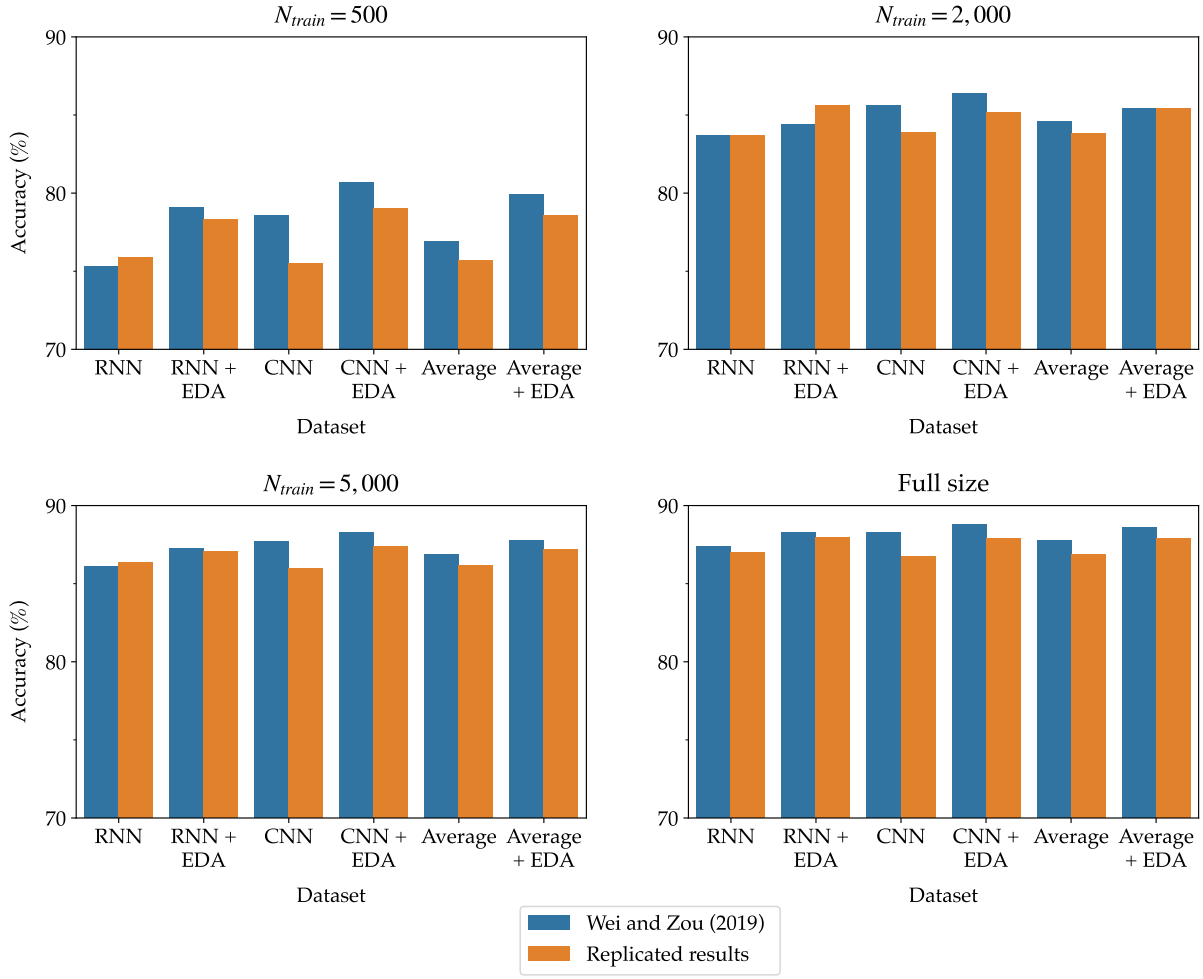

FIGURE 5.3: Comparison of replicated results to Wei and Zou (2019) for the average performances (%) of CNN and RNN models with and without EDA across the five classification tasks, for different sizes of the training datasets.

	Dataset size			
Model	500	2000	5000	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 5.9: Average performances (%) across five text classification tasks for models with and without EDA on different training set sizes. *Average* indicates an average between the accuracy values of the CNN and RNN models. Figure from Wei and Zou (2019).

	Dataset size			
Model	500	2000	5000	Full set
RNN	75.9	83.7	86.4	87.0
+EDA	78.3	85.6	87.1	88.0
CNN	75.5	83.9	86.0	86.8
+EDA	79.0	85.2	87.4	87.9
Average	75.7	83.8	86.2	86.9
+ EDA	78.6	85.4	87.2	**87.9**

TABLE 5.10: Replicated results: average performances (%) across five text classification tasks for models with and without EDA on different training set sizes. *Average* indicates an average between the accuracy values of the CNN and RNN models.

Table 5.9 shows accuracies from Wei and Zou (2019) for CNN and RNN models trained with and without applying standard EDA to the training corpora, while Table 5.10 shows the accuracies of the replicated results. A comparison of the results is shown in Figure 5.3. In the replicated results, the baseline RNN model achieves similar accuracies to those presented in Wei and Zou (2019), but the baseline CNN model differs in accuracy by -3.1%, for the $N_{train} = 500$ dataset. Overall, for the *Average + EDA* accuracies, the replicated values differ from Wei and Zou (2019) by -1.3% for $N_{train} = 500$, although the difference in accuracies is smaller for the full size dataset, differing by -0.7%.[1]

[1] In Wei and Zou (2019), it is unclear which alpha parameter is used to obtain the data for Table 5.9.

5.5.2 Different augmentation methods and alpha values

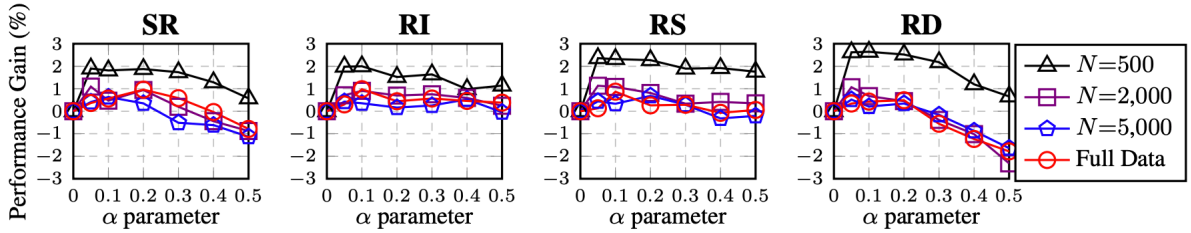

FIGURE 5.4: Average performance gain of EDA operations over five text classification tasks for different training set sizes. The α parameter is the proportion of words in a sentence changed by the augmentation method. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion. Figure from Wei and Zou (2019).

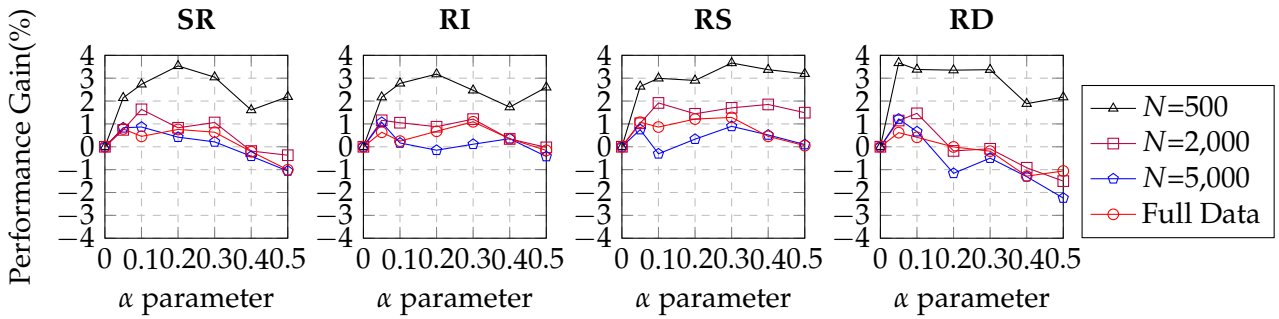

FIGURE 5.5: Replicated results: average performance gain of EDA operations over five text classification tasks for different training set sizes. The α parameter is the proportion of words in a sentence changed by the augmentation method. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion.

Wei and Zou (2019) next investigate the effect of individual EDA methods for both CNN and RNN models. Each method was applied to the $N_{train} = 500$, $N_{train} = 2,000$, $N_{train} = 5,000$, and full sizes of the datasets with a designated α value, where the α parameter is the proportion of words in a sentence changed by the augmentation method. Accuracy values for the CNN and RNN models were averaged. Averages were then taken across the five classification tasks for each α. Figure 5.4 shows the results from Wei and Zou (2019), and Figure 5.5 shows the replicated results.

Overall, the replicated results appear to confirm those of Wei and Zou (2019) for all dataset sizes except for $N_{train} = 500$. The replicated results show a similar peak at $\alpha = 0.05$ for the RD experiment, as well as decreasing performance gains for the four techniques as α decreases. However, in the replication, the maximum performance gains occur at different α values for the

SR, RI, and RS techniques. Additionally, the replicated results suggest that the maximum performance gain obtained by applying individual EDA methods is greater than that obtained by Wei and Zou (2019); the maximum y axis value required to display all data was 4%, compared to the maximum y axis value of 3% in Wei and Zou (2019).[2]

5.5.3 Different amounts of augmented training data

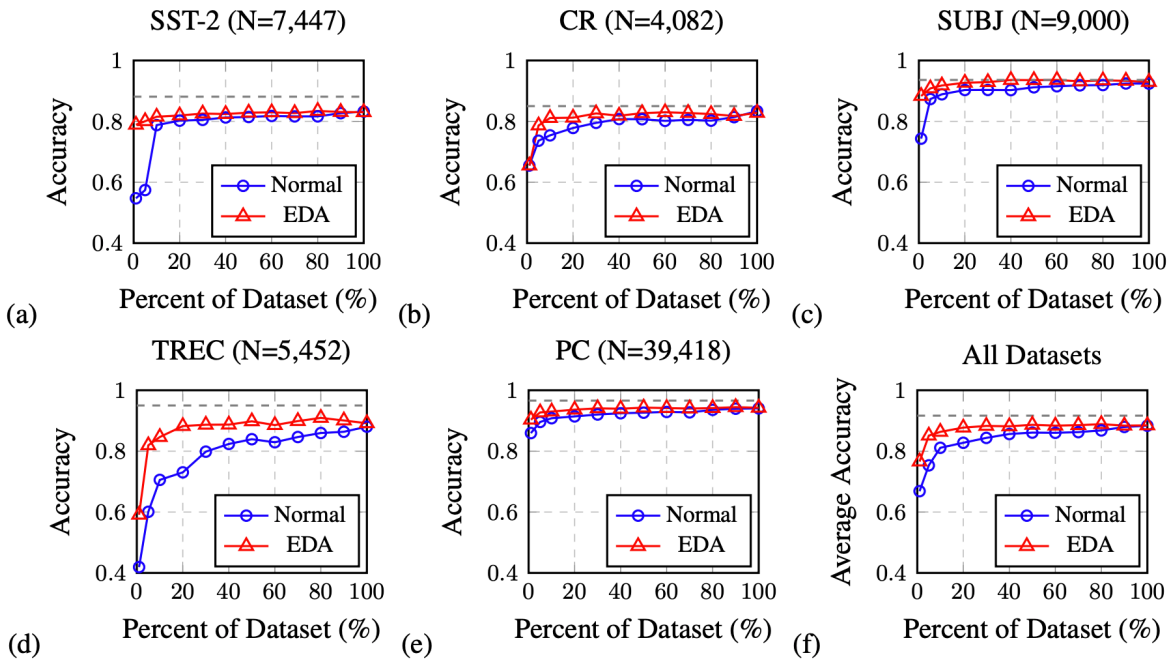

FIGURE 5.6: Performance on benchmark text classification tasks with and without EDA, for various dataset sizes used for training. Figure from Wei and Zou (2019).

[2]Due to time constraints, the performance gains of the replication are obtained from a single run of the experiment, generating one set of augmented training data. The replicated results may therefore not be a general trend.

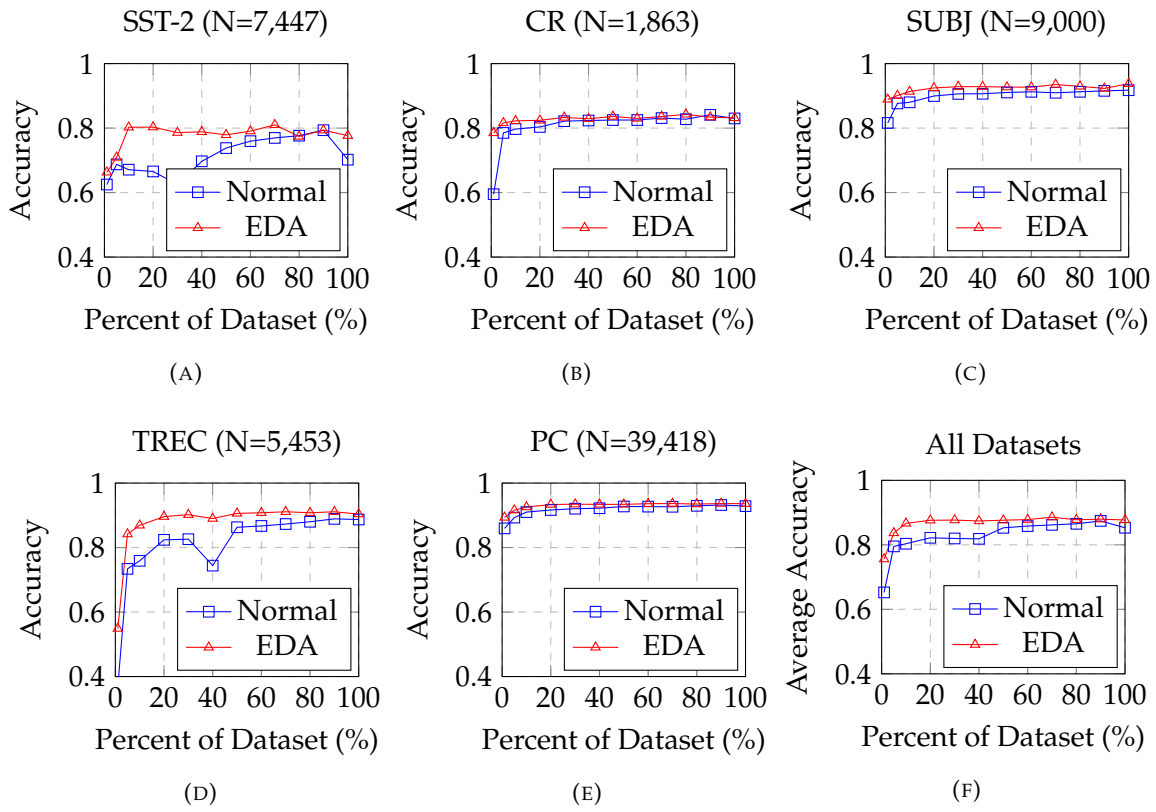

FIGURE 5.7: Replicated results: performance on benchmark text classification tasks with and without EDA, for various dataset sizes used for training.

This experiment examines the effect of standard EDA on the accuracy of CNN and RNN models trained on increments of the augmented training datasets for each task. Standard EDA was applied to the datasets before percentage increments of these datasets were taken and used for training. Accuracy values for the CNN and RNN models were averaged. The accuracies as obtained by Wei and Zou (2019) and the replication of the experiment are displayed in Figure 5.6 and Figure 5.7, respectively. The replicated results generally confirm those of Wei and Zou (2019); differences between the replicated results and those of Wei and Zou (2019) are likely caused by noise in training. In general, the results suggest that applying standard EDA to a corpus tends to yield higher accuracy on the classification tasks, especially for smaller increments of the augmented datasets. [3]

[3]The sizes of CR dataset (1,863 examples) and TREC dataset (5,453) used in the replication differ from the stated figures for the size of the CR and TREC dataset in Wei and Zou (2019) (4,082 and 5,452 examples, respectively). The source code provided by Wei and Zou (2019) used text files containing a total of 1,863 examples for the CR dataset, and 5,453 examples for the TREC dataset.

5.5.4 Different numbers of augmented sentences

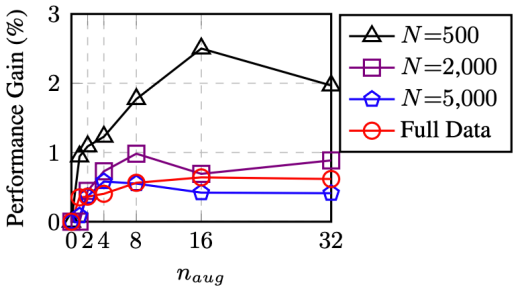

FIGURE 5.8: Average performance gain of EDA across five text classification tasks for various training set sizes. The parameter n_{aug} is the number of generated augmented sentences per original sentence. Figure from Wei and Zou (2019).

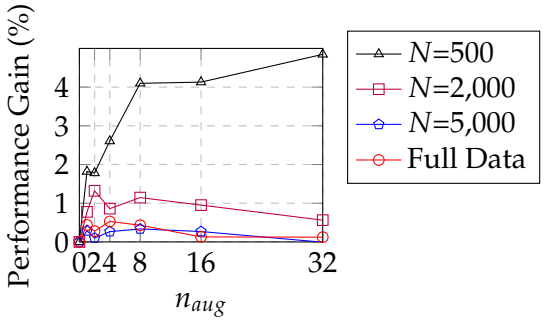

FIGURE 5.9: Replicated results: average performance gain of EDA across five text classification tasks for various training set sizes. The parameter n_{aug} is the number of generated augmented sentences per original sentence.

	n_{aug}	
N_{train}	Wei and Zou (2019)	Replicated results
500	16	32
2000	8	2
5000	4	8
More	4	4

TABLE 5.11: Recommended n_{aug} for EDA for Wei and Zou (2019) and replicated results.

This experiment investigates the effect of the number of augmented sentences generated per original training example, represented by the parameter n_{aug}. Figure 5.8 and Figure 5.9 summarize the results for the average performance gain of standard EDA when varying n_{aug}. Table 5.11 shows the recommended value of n_{aug} based on the average performance gain obtained

for CNN and RNN models, for both Wei and Zou (2019) and the replicated results. The replicated results differ in the optimal number of generated augmented sentences for $N_{train} = 500$, $N_{train} = 2,000$, and $N_{train} = 5,000$. Additionally, in the replication, the performance gain appears to decrease for values of n_{aug} greater than 8, for all dataset sizes other than $N_{train} = 500$.

5.6 Word length data augmentation

5.6.1 Augmentations using training corpus vocabulary

This experiment investigates the proposed length-based data augmentation technique in which words are replaced with other words of similar length from the training corpus vocabulary. For each training example, a fixed number of randomly selected words is replaced with a randomly selected word from the same training corpus, where the length of the replacement word length is within a certain number of characters of the length of the original word. The number of randomly selected words to replace in each training example is defined by a parameter α ranging from 0 to 0.5, where α is the proportion of words in a sentence changed by the augmentation method. For this experiment, the accuracies of CNN models are displayed in Figure 5.10, Figure 5.11, and Figure 5.12 for replacement words of length within $0, 1$, or 2 characters of the original word length, respectively.

Averaging across the tasks, the results of this experiment suggest that the length-based data augmentation method using words from the training vocabulary can increase the accuracy of CNN models, especially for smaller datasets and lower α values. However, the performance gains vary widely between the tasks and dataset sizes. SST-2 and TREC show the greatest performance gains from the length-based data augmentation. SUBJ and PC show more modest increases in accuracy, or even a decrease in accuracy for some dataset sizes. However, the models show performance losses on the CR task for all variations of the length-based augmentation, suggesting that length-based augmentation using the training corpus vocabulary may not be a suitable augmentation method for the CR classification task.

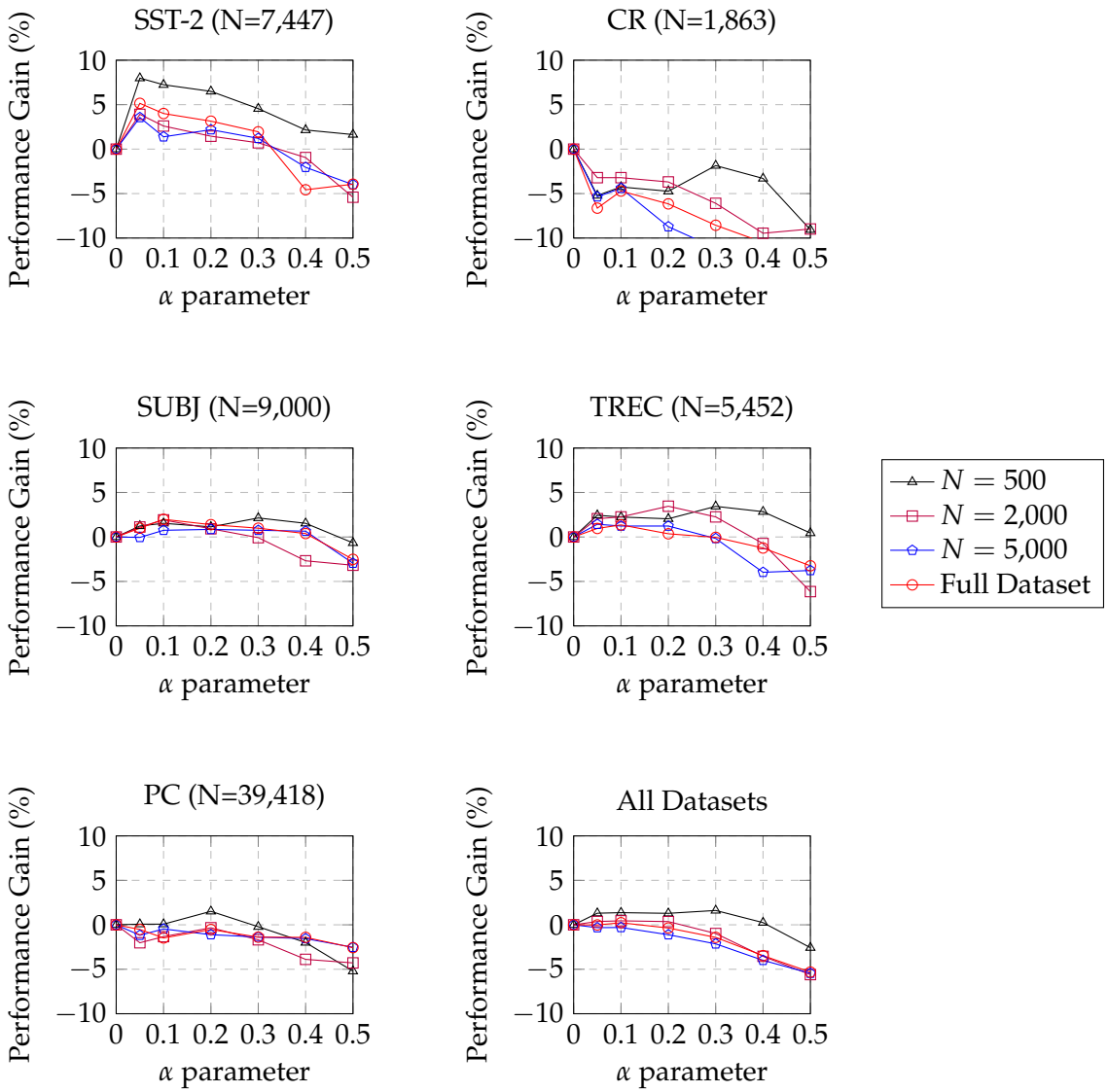

FIGURE 5.10: Average performance gain (%) for data augmentation using words from the training corpus vocabulary of length ±0 characters on the five classification tasks, for different alpha values. The α parameter is the proportion of words in a sentence changed by the augmentation method.

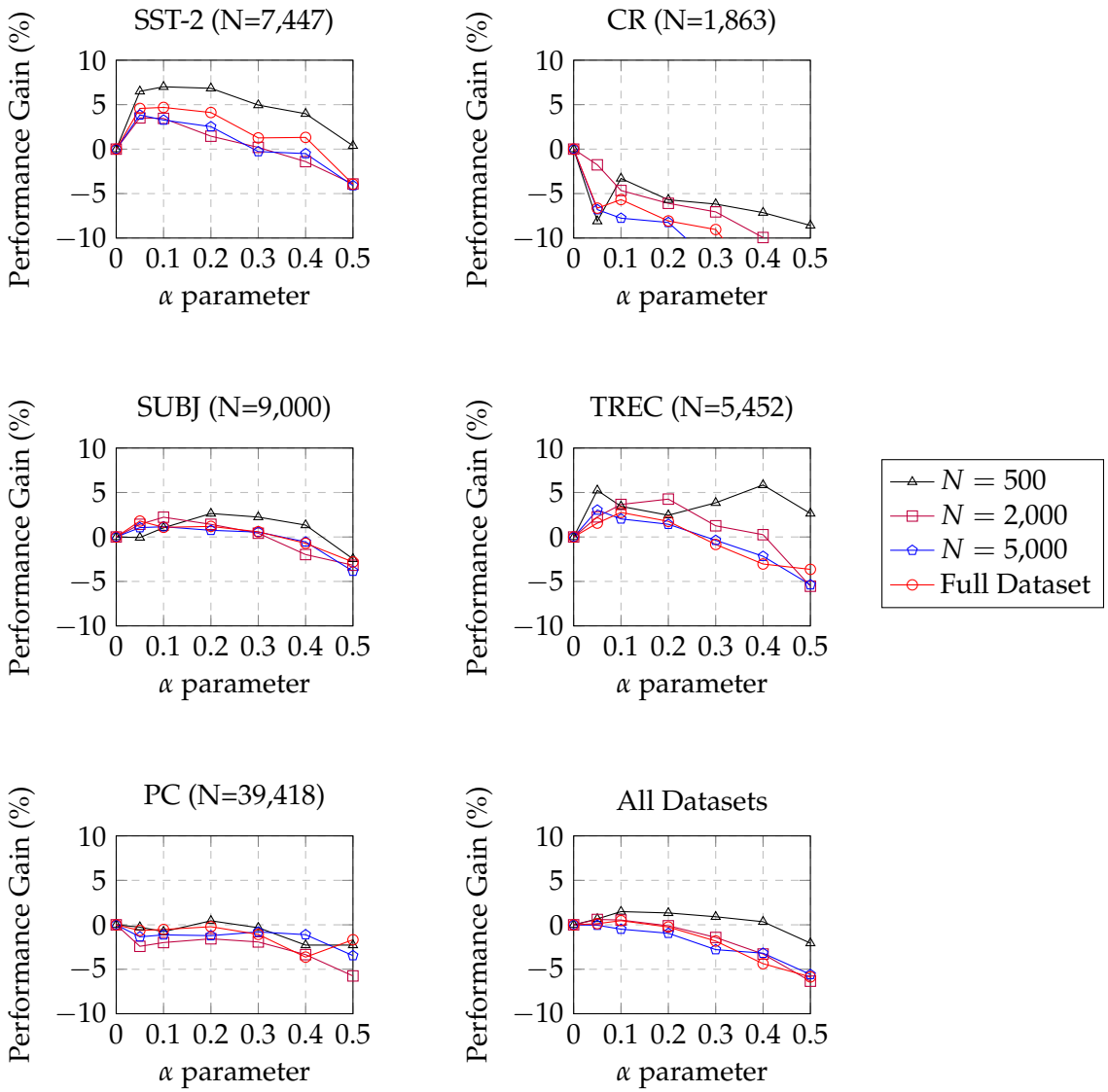

FIGURE 5.11: Average performance gain (%) for data augmentation using words from the training corpus vocabulary of length ±1 characters on the five classification tasks, for different alpha values. The α parameter is the proportion of words in a sentence changed by the augmentation method.

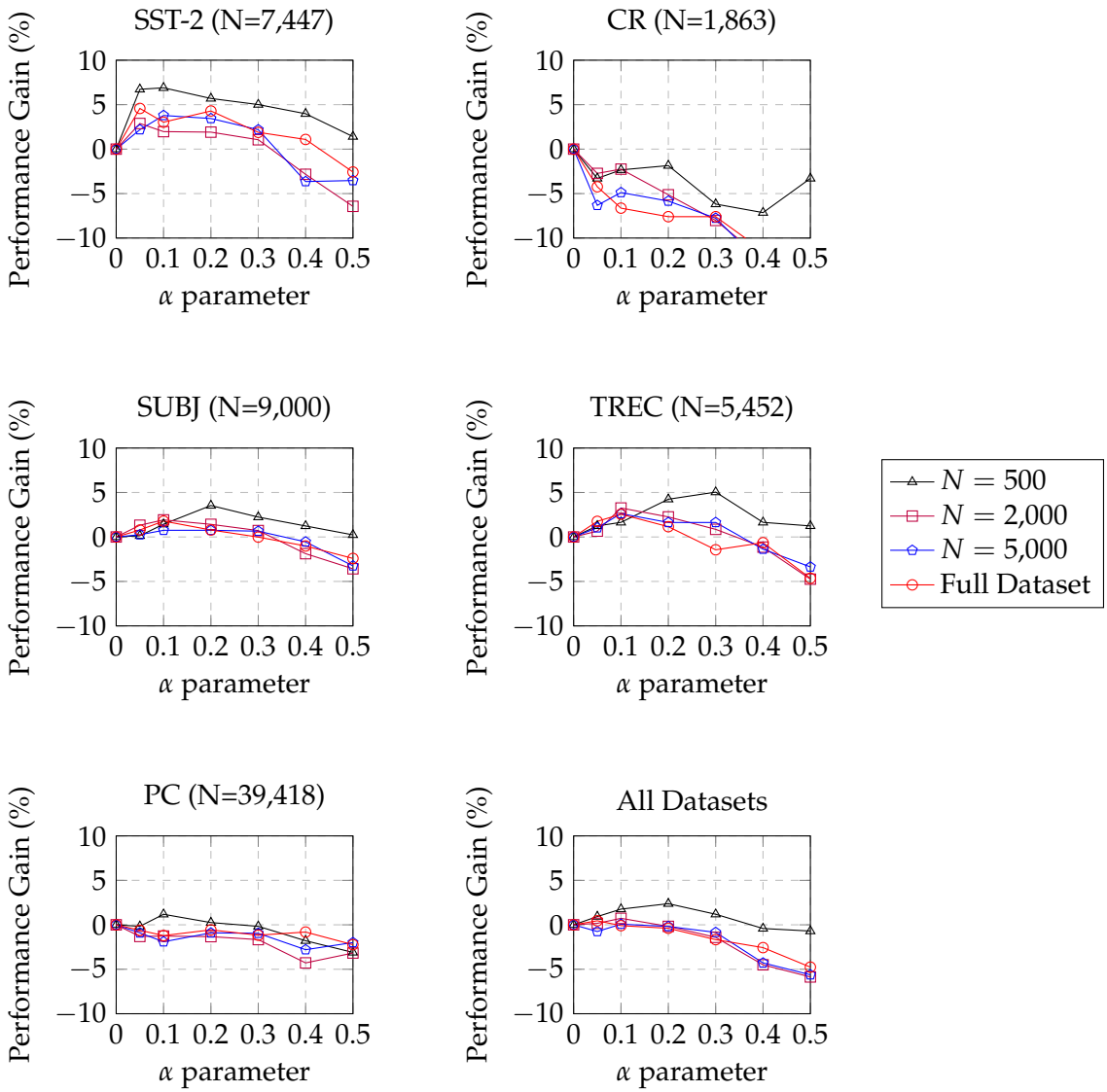

FIGURE 5.12: Average performance gain (%) for data augmentation using words from the training corpus vocabulary of length ±2 characters on the five classification tasks, for different alpha values. The α parameter is the proportion of words in a sentence changed by the augmentation method.

5.6.2 Augmentations using GloVe vocabulary

This experiment investigates the proposed length-based method of data augmentation, using replacement words from the GloVe vocabulary. Consistent with the previous experiment, the number of randomly selected words to replace in each training example is defined by a parameter α ranging from 0 to 0.5, where α is the proportion of words in a sentence changed by the augmentation. Due to time constraints, results are only shown for the GloVe experiment with

replacement words of the same length as the original word. These results are summarized in Figure 5.13.

Averaging across the tasks, the results of this experiment suggest that the length-based augmentation method using words from the GloVe vocabulary can increase the accuracy of CNN models, especially for smaller datasets. As with length-based augmentation using the training vocabulary, the performance gains differ between the tasks and dataset sizes. SST-2 and TREC again show the greatest performance gains from the augmentation method, although the α values at which maximum performance gains occur may differ compared to length-based augmentation using the training vocabulary. SUBJ and PC show more modest increases in accuracy compared to SST-2 and TREC. SUBJ and PC also appear to show more positive performance gain for length-based augmentation when using the GloVe vocabulary compared to using the training corpus vocabulary. As in the previous experiment, the models show performance losses on the CR task, despite using a different augmentation vocabulary. Overall, these results suggest that this length-based method may not be a suitable data augmentation technique for increasing performance on the CR classification task. This could potentially be due to the relatively small sizes of the dataset and vocabulary for the CR task compared to those of other datasets (Table 4.1). The replacement of words containing positive or negative connotation with non-synonym words can potentially distort the sentiment of the original example, generating augmented examples with neutral or opposite sentiment that are labeled with the original classification. The CR dataset may be more sensitive to the distortion effects of length-based data augmentation given the small number of training examples and size of the training vocabulary, potentially preventing the model from properly learning features that contribute to sentiment classification.

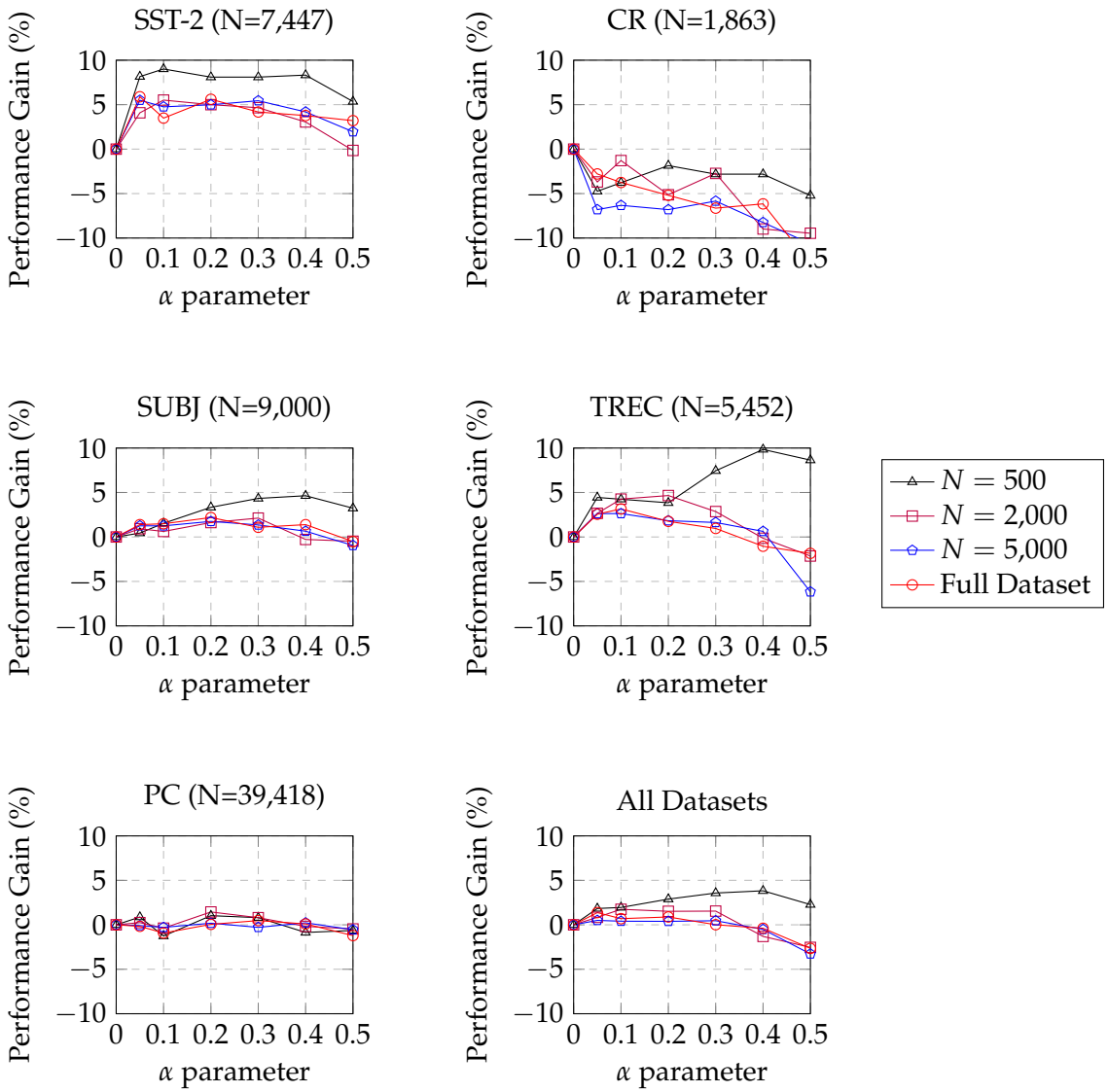

FIGURE 5.13: Average performance gain (%) for data augmentation using words from the GloVe vocabulary of length ±0 characters on the five classification tasks, for different alpha values. The α parameter is the proportion of words in a sentence changed by the augmentation method.

Chapter 6

Extensions and Future Work

6.1 Additional forms of data augmentation

One potential area of future direction would be to investigate whether the proposed length-based method of data augmentation is complementary to the Easy Data Augmentation Methods presented in Wei and Zou (2019). Additional work could include evaluating whether using length-based data augmentation in conjunction with Easy Data Augmentation yields further performance gains compared to Easy Data Augmentation alone.

Future work could also include investigation into other forms of data augmentation informed by linguistic knowledge. Additional experiments might include evaluating the effect on model performance of the following data augmentation methods, both individually and in conjunction with the Easy Data Augmentation:

1. Applying the synonym replacement technique used in Easy Data Augmentation, but restricting the set of candidate replacement synonyms to words with length within 0 to 2 characters of the original word length.

2. Inserting randomly selected words from a vocabulary such as the training corpus or GloVe vocabulary into a random location of the sentence. This differs from the random insertion method of Wei and Zou (2019), which inserts a random synonym of a randomly chosen word into a random location of the sentence a specified number of times.

3. Removing function words.

4. Changing a letter in a randomly selected word of each sentence. This form of data augmentation presents challenges, since the models used in my experiments are trained on the word level. Changing letters within words would yield non-words, which would likely not be compatible with the GloVe word embeddings. However, this method of data augmentation may be interesting to investigate for the character-level CNN models described in Zhang, Zhao, and LeCun (2015).

6.2 Additional text classification tasks

For consistency, I conducted experiments for the five benchmark classification tasks listed in Wei and Zou (2019): SST-2, CR, PC, SUBJ, and TREC. However, it is difficult to compare my results with other related work in the analysis of models and data augmentation for text classification tasks, which may use different datasets. Future direction for this research could focus on not only different benchmark datasets, but also datasets for different areas of text classification or datasets with different numbers of classes – four of the five tasks evaluated (SST-2, CR, PC, and SUBJ) focus on sentiment classification, all of which contain two possible classes for each example. The remaining task (TREC) is a question classification task, and is the only one of the five evaluated tasks to have greater than two possible classifications for a given example.

A dataset of interest might be SST-5 (Socher et al., 2013b), a 5-class variation of the Stanford Sentiment Treebank. For this dataset, each example is classified according to a more fine-grained sentiment scale, rather than simply positive or negative sentiment as in the SST-2 task. Using this dataset would help to determine the effect of changing the number of possible classes, while keeping the training data and goal of the task (sentiment classification) constant. It may also be of interest to investigate the datasets presented in Zhang, Zhao, and LeCun (2015) to enable fair comparison with closely related work.

6.3 Additional pretrained input representations

The pretrained GloVe word embeddings used in Wei and Zou (2019) and my experiments are constructed from cased data. One potential area of future direction could be to use uncased word embeddings such as the 42B token, 1.9M vocabulary size set of 300d Common Crawl GloVe embeddings provided by Pennington, Socher, and Manning (2014). However, this set differs from the set of word embeddings used in my experiments in both vocabulary size and number of tokens. Zhang, Zhao, and LeCun (2015) find in their analysis of character-level CNNs that not distinguishing between uppercase and lowercase letters in training data can increase accuracy for million-scale datasets, potentially due to a regularization effect. This regularization effect may also scale to the word-level, especially since I convert all data for the studied text classification tasks to lowercase during preprocessing.

Using word embeddings of dimension lower than the 300d vectors used in these experiment may also be an interesting extension. Comparison of the results for the CNN with one-hot vector representations and the CNN with random word embeddings showed that lower dimension word representations can have significantly positive effects on performance for some of the classification tasks. Pennington, Socher, and Manning (2014) also released several sets of pretrained GloVe word embeddings with dimension lower than 300d which could be used in future experiments.

6.4 Other models

All experiments involving neural models were conducted on CNNs (other than the replications of experiments from Wei and Zou (2019), which used both CNNs and RNNs). These experiments were limited to the CNN, as preliminary experiments demonstrated that the CNN obtained higher accuracy on the text classification tasks compared to the RNN. Additionally, time constraints on experimentation limited the amount of data available for the RNNs, which had longer training times compared to the CNN models. For future work, it would be interesting to examine the performance of other models such as RNNs to determine the different properties of these model architectures that contribute to performance on the classification tasks, especially since RNNs incorporate a temporal component. Another model to investigate could be the recurrent convolutional neural model proposed by Lai et al. (2015), which evaluates the recurrent convolutional model on the SST-5 dataset. It may be of interest to conduct similar studies for other classification tasks such as PC, CR, SUBJ, and TREC.

Chapter 7

Conclusion

The goal of this research has been to determine the features of model architecture and data augmentation that enable state-of-the-art models to obtain current performance on text classification tasks, and explore simple techniques of data augmentation to improve the performance of neural text classification models.

First, we present the task of text classification in natural language processing and examine the various traditional and deep neural methods used for classification. We also explore the motivation behind data augmentation and discuss various approaches taken by previous research for data augmentation as applied to natural language procecssing, such as synonym replacement. In particular, we focus on the method of Easy Data Augmentation, which uses four simple text editing techniques to generate new training examples, with a resulting positive effect on model accuracy for five selected classification tasks.

Next, we investigate features that can potentially contribute to the current state-of-the-art model performance for text classification. We analyze the performance of several traditional and deep learning models on a set of five classification tasks, to determine the effect of various properties such as model assumptions or architecture on classification performance. The investigated models range in complexity from a multinomial unigram Naive Bayes classifier to a convolutional neural network with trained GloVe word embeddings. In these experiments, we study the effects of different aspects of text classification models: the co-occurrence and word order assumptions for Naive Bayes classifiers, the lower dimension and trainability of word embeddings, and the latent linguistic knowledge encoded in pretrained word embeddings. Our results suggest that of these model attributes, the lower dimension of word embeddings and linguistic knowledge encoded in pretrained word embeddings have the greatest positive effect on model performance, although the effect of each of these factors is not uniform across the evaluated tasks.

We then turn to methods of data augmentation, analyzing the performance of current state-of-the-art models for previously explored methods of "easy" data augmentation as presented in previous work. Our results for this section confirm the findings of previous work. A novel method of simple data augmentation based on word length is presented, in which a number of words in a training example are randomly selected and replaced with a randomly selected

word of equal or similar length. This data augmentation technique is proposed because it does not require external linguistic knowledge, and word length closely reflects the average information content and conceptual complexity of words (Piantadosi, Tily, and Gibson, 2011; Lewis and Frank, 2016). Consequently, word replacement with another word of similar length can generate a new training example that is more similar to the original example, compared to a training example generated by using a replacement word of vastly different length. Our results suggest that this length-based method of data augmentation can significantly improve performance for some of the evaluated classification tasks. Finally, areas of future direction are presented, including variations on the dimensions of word embeddings, suggestions for other classification tasks on which to evaluate the models, and additional methods of simple data augmentation.

Appendix A

Appendix

A.1 CR dataset size

For all experiments, the size of my CR dataset (1,863 examples) differs from the stated figure for the size of the CR dataset in Wei and Zou (2019) of 4,082 examples. The source code provided by Wei and Zou (2019) only used text files containing a total of 1,863 examples.

A.2 Train and test splits

For the SST-2, PC, TREC, and SUBJ datasets, the provided train and test splits were used to train and test the models. For the CR dataset, I follow the method used in Wei and Zou (2019) to generate the train and test splits of the data, randomly taking 10% of the dataset for testing.

A.3 Stop words

A list of the stop words used when applying data augmentation is as follows: ["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "yourself", "yourselves", "he", "him", "his", "himself", "she", "her", "hers", "herself", "it", "its", "itself", "they", "them", "their", "theirs", "themselves", "what", "which", "who", "whom", "this", "that", "these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have", "has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but", "if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about", "against", "between", "into", "through", "during", "before", "after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over", "under", "again", "further", "then", "once", "here", "there", "when", "where", "why", "how", "all", "any", "both", "each", "few", "more", "most", "other", "some", "such", "no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", "can", "will", "just", "don", "should", "now", ""]

A.4 Training accuracies

	Dataset					
Dataset size	CR	SST2	SUBJ	TREC	PC	*Average*
500	61.4	68.9	62.2	30.0	59.7	56.4
2000	56.9	65.7	65.6	31.0	59.6	55.8
5000	52.5	64.8	87.4	32.4	73.9	62.2
Full set	53.0	67.2	81.9	35.4	86.2	64.7

TABLE A.1: Training accuracies (%) of the one-hot vector representation random word order CNN model on the five classification tasks, for different sizes of the training datasets.

	Dataset					
Dataset size	CR	SST2	SUBJ	TREC	PC	*Average*
500	63.6	65.6	57.3	28.3	62.0	55.4
2000	58.7	65.8	52.8	33.2	61.5	54.4
5000	55.6	65.8	86.3	26.1	75.9	61.9
Full set	52.8	65.1	82.1	35.2	89.8	65.0

TABLE A.2: Training accuracies (%) of the one-hot vector representation original word order CNN model on the five classification tasks, for different sizes of the training datasets.

	Dataset					
Dataset size	CR	SST2	SUBJ	TREC	PC	*Average*
500	76.4	66.5	82.6	89.2	90.8	81.1
2000	92.5	69.3	94.5	95.3	96.9	89.7
5000	95.7	71.6	96.9	96.4	96.4	91.4
Full set	95.4	68.4	97.8	96.6	97.0	91.0

TABLE A.3: Training accuracies (%) of the random word embeddings CNN model on the five classification tasks, for different sizes of the training datasets.

	Dataset					
Dataset size	CR	SST2	SUBJ	TREC	PC	*Average*
500	95.4	82.1	98.6	95.9	97.3	93.9
2000	96.4	95.6	98.5	97.9	96.8	97.0
5000	95.8	97.1	98.7	98.4	97.1	97.4
Full set	96.3	96.2	98.7	98.5	97.4	97.4

TABLE A.4: Training accuracies (%) of the untrained GloVe CNN model on the five classification tasks, for different sizes of the training datasets.

	Dataset					
Dataset size	CR	SST2	SUBJ	TREC	PC	*Average*
500	96.4	84.3	98.7	96.1	98.0	94.7
2000	97.1	97.1	98.8	98.0	97.2	97.6
5000	96.7	94.5	99.0	98.6	97.7	97.3
Full set	97.2	96.9	99.1	98.6	97.6	97.9

TABLE A.5: Training accuracies (%) of the trained GloVe CNN model on the five classification tasks, for different sizes of the training datasets.

Bibliography

Andreas, Jacob (2019). "Good-Enough Compositional Data Augmentation". In: *CoRR* abs/1904.09545. arXiv: 1904.09545. URL: http://arxiv.org/abs/1904.09545.

Ciresan, Dan Claudiu et al. (2010). "Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition". In: *CoRR* abs/1003.0358. arXiv: 1003.0358. URL: http://arxiv.org/abs/1003.0358.

Fadaee, Marzieh, Arianna Bisazza, and Christof Monz (July 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)*. Association for Computational Linguistics, pp. 567–573. DOI: 10.18653/v1/P17-2090. URL: https://www.aclweb.org/anthology/P17-2090.

Fauske, Kjell (2006). *Neural Network — TikZ Example*. https://texample.net/tikz/examples/neural-network/ [Accessed: February 26, 2021].

Ganapathibhotla, Murthy and Bing Liu (2008). "Mining Opinions in Comparative Sentences". In: *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*. COLING '08. Association for Computational Linguistics, 241–248. ISBN: 9781905593446.

Hu, Minqing 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. Association for Computing Machinery, 168–177. ISBN: 1581138881. DOI: 10.1145/1014052.1014073. URL: https://doi.org/10.1145/1014052.1014073.

Jurafsky, Daniel and James H. Martin (2009). *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc. ISBN: 0131873210.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1408.5882. arXiv: 1408.5882. URL: http://arxiv.org/abs/1408.5882.

Kobayashi, Sosuke (2018). "Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations". In: *CoRR* abs/1805.06201. arXiv: 1805.06201. URL: http://arxiv.org/abs/1805.06201.

Kolomiyets, Oleksandr, 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. Association for Computational Linguistics, 271–276. ISBN: 9781932432886.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (May 2017). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6, 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: https://doi.org/10.1145/3065386.

Lai, Siwei et al. (2015). "Recurrent Convolutional Neural Networks for Text Classification". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. AAAI Press, 2267–2273. ISBN: 0262511290.

Levy, Roger (2020). *Simple recurrent networks*. Lecture notes, MIT 9.19: Computational Psycholinguistics. Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology.

Lewis, Molly and Michael C. Frank (2016). "The length of words reflects their conceptual complexity". In: *Cognition* 153, pp. 182–195.

Li, Feifei, Ranjay Krishna, and Danfei Xu (2020). *Convolutional neural networks*. Lecture notes, CS231n: Convolutional Neural Networks for Visual Recognition. https://cs231n.github.io/convolutional-networks/. Department of Computer Science, Stanford University.

Li, Xin and Dan Roth (2002). "Learning Question Classifiers". In: *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*. COLING '02. Association for Computational Linguistics, 1–7. DOI: 10.3115/1072228.1072378. URL: https://doi.org/10.3115/1072228.1072378.

Liu, Qian et al. (2015). "Automated Rule Selection for Aspect Extraction in Opinion Mining". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. AAAI Press, 1291–1297. ISBN: 9781577357384.

Lu, Xinghua et al. (Sept. 2006). "Enhancing Text Categorization with Semantic-enriched Representation and Training Data Augmentation". In: *Journal of the American Medical Informatics Association : JAMIA* 13, pp. 526–35. DOI: 10.1197/jamia.M2051.

Miller, George A. (Nov. 1995). "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11, 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748.

Olah, Christopher (2015). *Understanding LSTM Networks*. https://tex.stackexchange.com/questions/494139/how-do-i-draw-a-simple-recurrent-neural-network-with-goodfellows-style [Accessed: March 26, 2021].

Pang, Bo 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. Association for Computational Linguistics, 271–es. DOI: 10.3115/1218955.1218990. URL: https://doi.org/10.3115/1218955.1218990.

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

Piantadosi, Steven T., Harry Tily, and Edward Gibson (2011). "Word lengths are optimized for efficient communication". In: *Proceedings of the National Academy of Sciences* 108.9, pp. 3526–3529. ISSN: 0027-8424. DOI: 10.1073/pnas.1012551108. eprint: https://www.pnas.org/content/108/9/3526.full.pdf. URL: https://www.pnas.org/content/108/9/3526.

Reading, Darren (2019). *Tex Neural Network*. https://github.com/dreading/tex-neural-network/commit/65dd41135f496344ff15de303175b067c04a1b3f [Accessed: February 26, 2021].

Socher, Richard et al. (Aug. 2013a). "Parsing with Compositional Vector Grammars". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 455–465. URL: https://www.aclweb.org/anthology/P13-1045.

Socher, Richard et al. (Oct. 2013b). "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1631–1642. URL: https://www.aclweb.org/anthology/D13-1170.

Summers, Cecilia and Michael J. Dinneen (2018). "Improved Mixed-Example Data Augmentation". In: *CoRR* abs/1805.11272. arXiv: 1805.11272. URL: http://arxiv.org/abs/1805.11272.

Wang, William Yang and Diyi Yang (Sept. 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*. Association for Computational Linguistics, pp. 2557–2563. DOI: 10.18653/v1/D15-1306. URL: https://www.aclweb.org/anthology/D15-1306.

Wei, Jason and Kai Zou (Nov. 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)*. Association for Computational Linguistics, pp. 6383–6389. URL: https://www.aclweb.org/anthology/D19-1670.

Xie, Ziang et al. (2017). "Data Noising as Smoothing in Neural Network Language Models". In: *CoRR* abs/1703.02573. arXiv: 1703.02573. URL: http://arxiv.org/abs/1703.02573.

Yu, Adams Wei et al. (2018). "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension". In: *CoRR* abs/1804.09541. arXiv: 1804.09541. URL: http://arxiv.org/abs/1804.09541.

Zhang, Hongyi et al. (2017). "mixup: Beyond Empirical Risk Minimization". In: *CoRR* abs/1710.09412. arXiv: 1710.09412. URL: http://arxiv.org/abs/1710.09412.

Zhang, Xiang, 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. MIT Press, 649–657.