Quantifying the Performance of Adversarial Training on Language Models with Distribution Shifts

Marwan Omar
University of Central Florida
Orlando, FL, USA
marwan@knights.ucf.edu

Soohyeon Choi
University of Central Florida
Orlando, FL, USA
soohyeon.choi@knights.ucf.edu

DaeHun Nyang
Ewha Womans University
Seoul, Republic of Korea
nyang@ewha.ac.kr

David Mohaisen
University of Central Florida
Orlando, FL, USA
mohaisen@ucf.edu

ABSTRACT

Adversarial training has recently emerged as an important defense mechanism to robustify machine learning models in the presence adversarial examples. Although adversarial training can boost the robustness of machine learning algorithms by a margin, research has not been conducted to determine if adversarial training is effective in the long-term. As deployments of machine learning algorithms are characterized by dynamics, change of the underlying model is inevitable. The dynamics are a result of model's evolution over time by introducing new training data and drifting the model by changing its parameters. In this paper, we examine the limitations of adversarial training due to the temporal changes of machine learning models. Using a natural language task, we conduct various experiments using a variety of datasets to measure the impact of concept drift on the efficacy of adversarial training. In particular, our analysis shows that certain adversarially-trained models are even more prone to the drift than others. In particular, WordCNN and LSTM-based models are shown more susceptible to the temporal changes than others such as BERT. We validate our findings using multiple real-world datasets on different network architectures. Our work calls for further research into the temporal aspects of adversarial training.

CCS CONCEPTS

• **Adversarial attacks**; • **Adversarial training**; • **Concept drift**; • **NLP Robustness**;

KEYWORDS

Sentiment analysis, robustness, concept drift, adversarial training

ACM Reference Format:
Marwan Omar, Soohyeon Choi, DaeHun Nyang, and David Mohaisen. 2022. Quantifying the Performance of Adversarial Training on Language Models with Distribution Shifts. In *Proceedings of the 1st Workshop on Cybersecurity and Social Sciences (CySSS '22), May 30, 2022, Nagasaki, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3494108.3522764

CySSS '22, May 30, 2022, Nagasaki, Japan

ACM ISBN 978-1-4503-9177-1/22/05…$15.00
https://doi.org/10.1145/3494108.3522764

1 INTRODUCTION

Recent research has demonstrated that Deep Neural Networks (DNN) in general, a popular family of techniques used in many Natural Language Processing (NLP) tasks, are vulnerable to adversarial attacks [6, 11, 14, 15, 26]. Adversarial attacks, also referred to as adversarial examples (AEs), are techniques that perturb the input data to force an NLP model to produce incorrect predictions [26, 27]. The fact that NLP models misclassify examples that are slightly different from a correctly classified input demonstrates a fundamental weakness in the training algorithms for NLP systems.

To address this issue in the learning models, extensive research has been conducted. One particularly active area of research in this space with a significant number of techniques is the *adversarial training*, first introduced by Goodfellow *et al.* for vision applications in [15]. In [15], the effectiveness of adversarial training is demonstrated by leveraging the gradient-based optimization technique. Moreover, it is shown that models can be made more resistant to AEs by training them on both the original and adversarial examples, thus the term is named as "*adversarial training*".

The general intuition behind the operation of adversarial training is that a model that is well-trained should perform reasonably well when it is tested with new (unseen) examples and resist adversarial attacks. To this end, several research works have proposed adversarial training techniques to defend against adversarial examples and eventually achieve the model robustness. A vast majority of the research studies addressing this problem take an NLP task (e.g., sentiment classification, Natural Language Inference (NLI), etc.), train an NLP model (e.g., BERT) with adversarial examples, and measure the model's prediction accuracy [19, 30, 32]. In doing so, they demonstrate that the adversarial training has improved model's accuracy significantly and they consider that as a contributing factor to the robustness against adversarial attacks.

A vastly unexplored aspect of this approach is that it considers a static view that does not deal with how learning methods would be deployed in reality, evolve over time, and be subject to distribution variations. In other words, those research works fall short in re-evaluating NLP models upon adversarial training to determine if they remain effective under different time settings.

The shift of models as a result of the distribution shift in the input data is intrinsic in the context of model training, as different machine learning algorithms are prone to the phenomena called "*concept drift*", which is well-documented in the literature. Concept drift makes the aforementioned shortcoming quite visible in an adversarially-trained model. In short, concept drift means that the statistical properties of the target variable (i.e., model's prediction accuracy) can change over-time in unforeseeable ways [23]. In the context of NLP models, concept drift means that a model's performance about NLP tasks can degrade over time as it sees new data [23]. This is a crucial aspect that deserves further attention, because NLP models could eventually make poor predictions on the data it has not seen before, while (distribution-wise) the data still are within the training fold.

Although *concept drift* is recognized and studied before, especially for pattern recognition [20, 28, 34], it has not been given any attention in the NLP domain, especially in the context of adversarial training. Motivated by this observation, we initiate the study of concept drift in NLP by evaluating their impact on the effectiveness of adversarial training. While it is intuitive that adversarial training will be affected by concept drift, *this study serves as a quantification of this impact with various learning models and datasets.*

Contributions. Equipped with this perspective, we make the following contributions. (1) Recognizing that adversarial examples are distribution-dependent, we stipulate that adversarial training is limited due to distribution shift in the underlying data. This distribution shift is best seen in model updates due to the arrival of new data observations. (2) We experimentally validate this shortcoming of adversarial training due to distribution shift on sentiment analysis, a popular natural language processing task, using various datasets and learning techniques. Among other interesting findings, we show that the degradation in the performance of adversarial training is both dataset and learning technique dependent.

Organization. The related work is outlined in §2. Background and methodology are outlined in §3. The evaluation is highlighted in §4, followed by concluding remarks and open questions in §5

2 RELATED WORK

The machine learning (ML) literature has many research studies [5, 8–10, 24] advocating and demonstrating the importance of adversarial training as a defense mechanism against adversarial attacks following the work of Goodfellow *et al.* [15] which showed that adversarial training can withstand adversarial attacks and thereby make such models more robust against real-world attacks.

In the NLP domain, adversarial training has almost become a *de facto* standard to achieve robustness to adversarial attacks. In [30], Wang *et al.* present a Controlled Adversarial Text Generation (CAT-Gen) model that, given an input text, generates adversarial texts through controllable attributes that are known to be irrelevant to task labels. Their proposed model consists of an encoder and a decoder for text generation, and a module network that encodes the information of controllable attributes and generates adversarial attacks by changing the controllable attributes.

Hendrycks *et al.* [17] studied the effect of pretaining ML models on robustness in the image domain, showing improvements of ~10% in robustness using adversarial examples and out-of-distribution detection. This work falls short in discussing the long-term stability of ML models and how the performance might degrade over-time as models evolve and are introduced to future data.

Original
Perfect performance by the actor → Positive (99%)
Adversarial
Spotless performance by the actor → Negative (100%)

Figure 1: An adversarial example generated using [19] TextFooler for a BERT-based sentiment classifier. Swapping out "perfect" with synonym "spotless" completely changes the model's prediction, even though the underlying meaning of the text has not changed

Madry *et al.* [24] studied the adversarial robustness of deep neural networks from a robustness optimization perspective. They explored the link between neural network architecture and adversarial attacks, and concluded that model capacity plays a role in defending against adversarial examples. They also demonstrated that adversarial training can yield more robust models to adversarial attacks. This study did not indicate whether the same optimization technique applies to other ML models such as NLP models.

Iyyer *et al.* [18] present a framework called Syntactically Controlled Paraphrase Network (SCPN) for generating adversarial examples using an encoder-decoder model for adversarial training data generation. Their technique, however, assumes that the model will be static, meaning that the mapping learned from adversarial training data is just as valid in the future on new data and that the model will change in the future.

Zhu *et al.* [33] propose an adversarial training algorithm, FreeLB, that promotes higher invariance in the embedding space. To validate the proposed approach, they apply it to Transformer-based models for natural language understanding and commonsense reasoning tasks. Experiments on the GLUE benchmark show that when applied only to the fine tuning phase, it boosts the overall performance of BERT model from 78.3% to 79.4%, and RoBERTa-large model from 88.5% to 88.8%. The work, however, does not consider the temporal aspects of model training.

Wong [31] introduced DANCin SEQ2SEQ, a framework for generating adversarial examples targeting black-box classifiers focusing on a binary classifier for spam detection using the spam-ham dataset by re-framing the adversarially-generated text as a reinforcement learning technique and drawing on GAN-inspired learning techniques to generate examples attacking a targeted classifier.

Recent work addressed the effect of concept drift over the long-term performance of ML models. Bichine *et al.* [7] conducted a comprehensive study on the effect of concept drift using a tweeter stance detection model as the classification task and concluded that the performance of ML classification task degrades over time as ML algorithms are introduced to new data from the real world due to concept drift. They also indicated that ML models (especially classification tasks) should implement adaptation approaches in order to cope with concept drift issue. Their work, however, does not consider how such a shift affects adversarial training.

3 BACKGROUND AND METHODOLOGY

In this section, we review the background and the methodology of experiments outlined in the rest of this paper. To initiate our

discussion, we review the main natural language task of interest, namely sentiment analysis. We then outline various definitions of adversarial examples and adversarial training, followed by our implementation to examine the impact of varying models on the performance of adversarially-trained sentiment analysis tasks.

3.1 Sentiment Analysis

While adversarial examples are applicable on a large number of tasks, we use sentiment analysis (SA) as our task of choice, given the prevalence of this task in general. SA is a text classification task that is often used in the context of analyzing text and natural language data [1–4]. SA works by assigning weighted sentiment scores to the topics and categories within a sentence. The purpose of SA in the context of NLP is to identify, extract, and analyze subjective markers of data coming from textual sources such as the opinions of customers in the Amazon Review dataset, or the opinions of viewers in the IMDB dataset. In this way, SA uses NLP to identify the sentiments of users on a given topic. When we apply SA techniques to NLP, we can classify a given input text by the polarity of its sentiment as being positive, negative, or neutral [12].

Definition: Sentiment Analysis. *Let $x \in \mathcal{X}$ be an instance of input text, where $\mathcal{X}$ is the space of the input text, and y be the target label of sentiment (e.g., for binary sentiment; positive vs. negative, y could be either 0 or 1; in the general case, $y \in \mathbb{R}^k$, where k is the number of classes being predicted). The sentiment classification task is denoted as a mapping function $h_\theta : \mathcal{X} \rightarrow \mathbb{R}^k$ (alternatively, $h_\theta : x \rightarrow y$).*

As can be seen from the above definition, SA can be implemented using a variety of algorithms to realize h_θ, both deep and shallow, including BERT, WordCNN, LSTM, etc., which we use in this work.

In realizing h_θ, we define a loss function ℓ, where the purpose is to minimize $\ell(h_\theta(x), y)$. Given that a model is trained on more than one input sample (e.g., m of them), the task then becomes to minimize the average loss across those samples. That is,

$$\min_\theta \frac{1}{m} \sum_{i=1}^{m} \ell(h_\theta(x_i), y_i) \tag{1}$$

3.2 Adversarial Examples

Adversarial examples are inputs that are designed to attack and fool an ML system. More specifically, an adversarial example is defined as an input x' obtained by solving the following optimization:

$$\max_{x'} \ell(h_\theta(x'), y) = \max_{\delta \in \Delta} \ell(h_\theta(x + \delta), y) \tag{2}$$

That is, by solving this optimization we try to find x' defined as $x + \delta$ for some perturbation δ obtained from the allowed set of perturbations, Δ, such that the loss between the prediction by h_θ and the original label, say y, is maximized. We notice that $\delta \in \Delta$ is typically dependent on x's distribution.

In this work, we are only interested in adversarial examples on the textual inputs for the sentiment application. We notice that there has been several studies addressing the optimization problem in (2). In the following, we consider one such work for highlighting the background and the key elements employed in our analysis, although this part can be replaced by any adversarial generation algorithm as it is a secondary aspect to the contribution.

Adversarial Examples. Jin *et al.* [19] introduced simple yet efficient baselines for realizing adversarial examples for BERT models in the block-box threat model. Moreover, they demonstrated how these adversarial examples can be used to attack NLP models, particularly the sentiment analysis task, and ultimately measured their robustness against adversarial attacks. Generally, to generate an adversarial example that meets the definition shown in (2) for the sentiment analysis task, a combinatorial optimization problem is solved using various heuristic search algorithms [25]. In other words, an adversarial example generation algorithm will search for inputs that produce incorrect outputs leading to a misclassification.

In Figure 1, we show an adversarial example generated using TextFooler, due to Jin *et al.* [19] for the BERT-based sentiment classifier. The algorithm works in two major steps. First, starting with a sentence of n words, $x = w_1, w_2, \ldots, w_n$, the algorithm ranks those words based on their importance. Noting that only the important words in x are going to contribute more significantly to the outcome of h_θ, the authors argue that changing those important words will significantly affect the output of $h_\theta(x)$. Moreover, given that the modification will only affect those important words, it will, by definition, be minimal. Once the important words are selected, the second step of *word transformer* is invoked, where words are replaced with other words that have similar semantics, fit well in the context, and force the target model to misclassify the input sentence x. Upon executing the word transformer step, various sanity checks are imposed, including part-of-speech enforcement and semantic similarity check. The example in Figure 1 shows a sentence (original and adversarial example) resulting from the execution of this heuristic: replacing "perfect" with the synonym "spotless" fools the model, leading to an incorrect prediction. We note that the underlying meaning of the original example has not changed, thanks to the constraints over the adversarial example.

3.3 Adversarial Training

Adversarial training is a technique used to improve the robustness of deep network models by integrating adversarial examples into the process of the training of the model [22]. For NLP tasks, adversarial training refers to the process of creating adversarial examples, by either re-training a model on adversarial examples that have successfully attacked and fooled the model or by incorporating input perturbations into the model training, e.g., using the approach highlighted in the previous section. In either case, adversarial training is performed by adversarially perturbing the text embedding space. Technically, adversarial training aims to (adversarially) optimize the loss function to minimize the computational cost associated with creating an adversarial example. In other words, we try to modify the training objective by applying a small perturbation to the input text that maximizes the adversarial loss as the following.

Definition: Adversarial Training. *We define $J(h_\theta)$ as a cost function (risk, or loss under the true distribution of samples) and write down this risk function formally as:*

$$J(h_\theta) = \mathbf{E}_{(x,y)\in\mathcal{D}}[\ell(h_\theta(x), y)] \tag{3}$$

where ℓ is the loss function, $h_\theta(x)$ is the predicted output when the input is x, $\mathcal{D}$ is the empirical distribution over samples, $\mathbf{E}$ is the expectation function, and y is the target output. Since we often do

not know the actual distribution of the data, we approximate the distribution by considering a set of samples $D = \{(x_i, y_i) \sim \mathcal{D}\}; i = 1, \ldots, m$, *from which the empirical cost is calculated as*

$$J'(h_\theta, D) = \frac{1}{|D|} \sum_{(x,y)\in D} \ell(h_\theta(x), y) \tag{4}$$

In the training process, we attempt to minimize $J'(h_\theta, D_x)$, *where* D_x *is a training set. Using the same notation above, one can generalize the cost into an adversarial cost, which we define as:*

$$J_{\text{adv}}(h_\theta) = \mathbf{E}_{(x,y)\in\mathcal{D}}[\max_{\delta\in\Delta} \ell(h_\theta(x+\delta), y)] \tag{5}$$

Similar to the rationale highlighted for (4), we define the empirical adversarial cost as:

$$J'_{\text{adv}}(h_\theta, D) = \frac{1}{|D|} \sum_{(x,y)\in D} \max_{\delta\in\Delta} \ell(h_\theta(x+\delta), y) \tag{6}$$

In the adversarial training, $J'_{\text{adv}}(h_\theta, D)$ in (6) is minimized.

Implementation Details. In our analysis, we followed [33] and utilized the inner ascent steps of the Projected Gradient Descent (PGD), a popular and powerful optimization algorithm for machine learning. In this method, the gradients of the parameters can be obtained with almost no overhead when computing the gradients of the inputs. Moreover, we implemented the algorithm using the TextAttack framework [26] to test the AEs we generated.

3.4 Distribution Shift and Associated Questions

In our description of the training process so far, for both the regular model training using the optimization in (1) and the adversarial training using (6), we assumed a static dataset, D. Even when training adversarially, perturbation δ is going to be defined in terms of the dataset D, limiting the impact of the adversarial training to the current data, and making the adversarial examples less relevant to other datasets. In the actual deployments of NLP tasks, such as sentiment analysis, however, the underlying model will be constantly exposed to new data. The newly introduced data may even be out of distribution (OOD) with respect to the original data used for training and evaluating the performance of the model (baseline; before deployment). Moreover, the distribution of the data will change over time, causing temporal changes to the model and causing distribution (or even concept) shift/drift [13].

Concept drift, for instance, implies that statistical properties of the target variable (e.g., model's prediction accuracy; conversely, loss) can change over-time in an unforeseeable ways [23]. In turn, this causes the NLP model to make incorrect predictions (i.e., variable target) even though the distribution of the input may stay constant. Moreover, a model's performance can degrade over time as it sees new data [23]. This is a crucial aspect that deserves attention because NLP models could eventually make poor or even wrong predictions on data it has not seen.

Starting with a model training used to achieve the objective in (1), one expects an accuracy (possibly high, since the loss is minimized as part of this optimization). Introducing an adversarial example, as defined in (2), one would expect the loss $\ell(h_\theta(x+\delta), y)$ to be significantly increased (model under attack). To cope with the high loss, one could imagine an adversarial training step, by solving the optimization in (6). We note that the adversarial examples are obtained using D, an empirical distribution using a set of finite examples (x_i, y_i) for $i = 1 \ldots m$. The question then becomes: how would the loss of this model against adversarial examples in $J'_{\text{adv}}(h_\theta, D)$ be impacted if D is changed to D', updating h_θ without updating $J'_{\text{adv}}(h_\theta, D)$? In other words, by solving $\min_\theta J'_{\text{adv}}(h_\theta, D)$ for some D, then updating the dataset D to D', and solving for θ' in $\min_\theta J'(h_\theta, D')$, how big is the gap in the inner model loss? This is, how big is the gap computed as the following difference:

$$\frac{1}{|D'|} \sum_{(x,y)\in D'} \ell(h_{\theta'}(x), y) - \frac{1}{|D|} \sum_{(x,y)\in D} \ell(h_{\theta'}(x+\delta), y)$$

Answering the question might be straightforward for a binary answer (i.e., whether there will be a gap or not, since it is intuitive that the loss will increase and the performance will degrade when tested against old adversarial examples generated by taking D into account). However, our focus in this work is rather the quantification of this gap, which is both interesting and important.

Addressing this question is the key contribution of this study. Namely, we are interested in exploring the interplay between this drift (of the original model) and the performance of adversarial training, potentially as the model gets retrained. We do that empirically to verify and validate the importance of distribution (concept) drift for NLP models using real-world datasets and a variety of learning models and architectures; transformers-based, LSTM, and WordCNN-based models, which are all widely used in the literature.

The flow of our experimental setup starts by a pretrained model, exposed to adversarial examples upon which an adversarial training is conducted to cope with those adversarial examples. The concept drift (of the original model) takes place, and model retraining is conducted so as to maintain the accuracy of the original model. Upon doing that, we explore whether the adversarial retraining is still effective in addressing the same adversarial examples already built in the model through the adversarial training.

4 RESULTS AND DISCUSSION

In the following, we present our experiments conducted on various datasets using different model architectures and demonstrate that adversarial training is in fact limited due to distribution shifts in the data used in the underlying model and associated retraining.

4.1 Datasets, Algorithms, and Metrics

4.1.1 Datasets. As shown in Table 1, we conducted our experiments using three datasets: IMDB, Movie Review, and Yelp Polarity. We use those datasets in this study because they are standard datasets (benchmarks) that are popular and widely used in the domain of sentiment analysis tasks and classification in general.

4.1.2 Algorithms. A range of deep learning algorithms that have been shown to provide state-of-the-art classification results for sentiment analysis are used: BERT, WordCNN, and LSTM.

BERT. Bidirectional Encoder Representations from Transformers (BERT) is a transformer-type deep neural network which has been fine-tuned and pre-trained using a gigantic dataset. The pre-trained characteristic of BERT makes it a perfect fit for various NLP tasks, such as sentiment analysis, question answering, and paraphrasing,

Table 1: Datasets: IMDB, Movie Review, and Yelp for our sentiment analysis task. Each dataset has binarised ratings and is set as positive and as negative, and split into a training and test set.

Dataset Name	Dataset Description	Atributes
IMDB	Large Movie Review Dataset	set of 25,000 for training, and 25,000 for testing
MR	Movie Review Dataset	set of 5,331 for training, and 5,331 for testing
Yelp	Large Yelp Review Dataset	set of 560,000 for training, and 38,000 for testing

among others. Moreover, BERT allows its adopters to build highly-complex and precise language understanding models, which could be used for different NLP applications [21]. On the other hand, the fining-tuning capability provided by BERT allows for embedding it into various models architectures and thus efficiently fine-tuning a model to become robust to adversarial attacks.

WordCNN. Word Convolutional Neural Network (WordCNN) is a neural network used for linguistic tasks, such as text classification. For the wordCNN model to effectively classify text, it implements a word embedding layer and a one-dimensional convolutional network [16]. The word embedding layer is essential for the wordCNN model to handle the challenges associated with addressing NLP tasks including text classification. Additionally, the word embedding layer is also crucial for fitting the CNN model and ultimately achieving high performance on the given linguistic task.

LSTM. The Long-Short Term Memory (LSTM) is a particular type of recurrent neural network (RNN) capable of learning and handling sequential data using a recurrent layer or cell. The recurrent layer or cell of the LSTM network enables them to memorize data for long periods of time. This is done by producing its own output at a particular time stamp from part of the input to the next time stamp. The ability of LSTMs to remember long-term dependencies makes LSTMs especially useful in addressing the long-term dependency problem associated with other RNNs [29].

4.1.3 Evaluation Metrics. For our evaluation, we use two metrics: the accuracy and misclassification rate. The accuracy is calculated as the number of correct sentiment predictions made by the model normalized by the total number of predictions. The misclassification rate is calculated as the total number of the wrong predictions made by the model normalized by the number of predictions.

4.2 Results

4.2.1 Baseline. We implement the above three algorithms for establishing a baseline of performance (using the accuracy) of the learning algorithms without any attacks. As we can see from Table 2, BERT consistently achieves the highest classification accuracy (90.01%, 87.03%, and 96.12% on the three datasets, respectively). Conversely, WordCNN consistently achieves the lowest classification accuracy (86.32%, 79.41%, and 92.29% on the three datasets, respectively). The fact that BERT outperforms the other two models under different datasets is expected as BERT is a state-of-the-art model and this is consistent with findings in the literature.

4.2.2 Misclassification Rate Under AEs. We study the impact of the adversarial examples on the performance of the different models using the different datasets by employing the adversarial example generation method mentioned in section 3.2. For each run, we use 1, 000 adversarial examples.

Table 2: Baseline experiment shows the classification accuracy of each of the three models prior to adversarial training. Note that BERT consistently achieves the highest classification accuracy. Conversely, WordCNN consistently achieves the lowest classification accuracy.

Dataset	Model	Accuracy
IMDB	BERT	90.01
	WordCNN	86.32
	LSTM	88.32
MR	BERT	87.03
	WordCNN	79.41
	LSTM	80.71
Yelp	BERT	96.02
	WordCNN	91.36
	LSTM	92.22

Table 3 shows that the misclassification rate upon introducing the adversarial examples is significantly high. Namely, it is shown that the BERT model exposed to adversarial examples suffers a misclassification rate of 96.31% when tested with the MR dataset. To further validate the distribution shift of models and prove the transfer ability and generalization of our findings, we extend our experiments to different model architectures and observe that the WordCNN and LSTM-based models suffer even a higher misclassification rate. In particular, LSTM-based model fared the worst with a misclassification rate of 99.71% (even after adversarial training) when tested with adversarial examples using the Yelp dataset.

Table 3: Misclassification under adversarial examples. The misclassification rate increased after introducing the adversarial examples. We observe that BERT is consistently performing better than other models across multiple datasets.

Dataset	Model	Misclassification Rate
IMDB	BERT	96.04
	WordCNN	97.89
	LSTM	99.22
MR	BERT	96.31
	WordCNN	97.92
	LSTM	99.62
Yelp	BERT	96.26
	WordCNN	97.95
	LSTM	99.71

4.2.3 Retraining with Adversarial Examples. To generate AEs we utilize the idea of Adversarial Text Generation by [30], although using a different model architecture and for different datasets.

Implementation Details. The adversarial examples generation model includes an encoder and a decoder for generating adversarial examples. The encoder and decoder are trained over a large text corpus to ensure that adversarial examples adhere to the linguistic

Table 4: Adversarial training results. Note that the accuracy rate, after adversarial training, for each model consistently increased across multiple datasets.

Dataset	Model	Accuracy
IMDB	BERT	93.24
	WordCNN	89.22
	LSTM	88.01
MR	BERT	89.98
	WordCNN	82.34
	LSTM	82.35
Yelp	BERT	98.12
	WordCNN	92.29
	LSTM	93.68

constraints and preserve semantics. For semantic preservation, we follow [25] and tighten the thresholds on the cosine similarity between embeddings of swapped words and the sentence encoding of original and perturbed sentences. We ensure and enforce the grammaticality of the adversarial examples by validating perturbations with a grammar checker. Moreover, we apply the semantics as well as the grammatical constraints at each step of the search following [26]. We conduct our experiments on real-world NLP datasets to demonstrate the effectiveness, applicability and generalizability of our approach. We show that our generated attacks are more diverse and more robust against model re-training and various model architectures. For the retraining, we adopted our three learning models to ensure the generalizability and transferability of our results to different network architecture and under multiple datasets.

As shown in Table 4, retraining models with adversarial examples lowers the misclassification rate and improves the accuracy, which is consistent with the literature of adversarial training. In Table 4, we present the results of retraining each of the the three models (BERT, WordCNN, and LSTM) with adversarial examples training. Specifically, we divide generated adversarial examples into two subsets, one is used for augmenting the training data, and the other is a hold-out set used for testing. With the augmented training data, we observe that adversarial training achieves accuracy gains consistently across all models and under different datasets.

4.2.4 Adversarial Examples with Training Model to Address Temporal Changes. Finally, we examine the impact of model retraining on the performance of retrained models using adversarial example. To simulate this idea, we have used the same datasets highlighted earlier, but for model retraining, we held a fold out (half of the dataset). Upon introducing retraining the models built with the adversarial training, with this held dataset, we obtain the results in Table 5. From these results, we observe that, generally, the performance of the adversarially-trained model degrades significantly (comparing Table 4 to Table 5). For example, we notice that the misclassification, while not as high as that in Table 3, is still as high as 91% (LSTM over MR) and is as low as 58% (BERT over IMDB). These results, in a way, confirm that while the performance of the model against adversarial examples reduced, the effect of adversarial training is still present although with very little impact defending against the introduced adversarial examples upon retraining.

Table 5: Misclassification rate with retraining upon adversarial training. Note that retraining models with adversarial training examples yields a lower misclassification rate.

Dataset	Model	Misclassification Rate
IMDB	BERT	58.17
	WordCNN	69.12
	LSTM	72.45
MR	BERT	84.25
	WordCNN	89.77
	LSTM	91.34
Yelp	BERT	80.22
	WordCNN	85.68
	LSTM	90.01

5 CONCLUSIONS AND OPEN DIRECTIONS

In this paper, we show that adversarial training has a very limited effect on NLP model's robustness due to the concept drift phenomena addressed by model retraining. We demonstrate that the performance of adversarial training significantly degrades over time through experiments on three real-world datasets (IMDB, MR, and Yelp) using three different model architectures (BERT, WordCNN, and LSTM). We observe that the misclassification rate of models with adversarial training is just as high as models without adversarial training. For instance, our BERT model trained with adversarial examples suffers a misclassification rate of 96.31% when tested with a new dataset (MR dataset).

To validate the temporal changes of models and prove the transferability and generalization of our findings, we extend our experiments to other model architectures and observe that WordCNN and LSTM-based models suffer even a higher misclassification. In particular, LSTM-based model fared the worst with a misclassification rate of 99.71% (after adversarial training) when tested with adversarial examples using the Yelp dataset.

The findings in this work call for further investigation into the impact of actual deployment on the behavior of models with respect to defense mechanisms. While we address adversarial training in this work, the insight can be applied to any defense mechanism that incorporates training processes, where the effect of this training diminishes by introducing new samples into the original model that may feature distribution shift. Given the simplicity of the model output we considered here (sentiment), it is unclear how much of our insight generalizes to other NLP tasks, e.g., Natural Language Inference and Question Answering, which will be our future work. Moreover, given the fundamental characteristics exploited in this work for demonstrating the limitations of adversarial training, it would be interesting to see how such insight translates to performance degradation in less constrained domains (e.g., image).

In conducting our experiments, we considered a simplistic scenario: the distribution shift of the model as result of the shift in the underlying data distribution is done once. This somewhat deviates from the natural data dynamics: a model is typically updated by data increments happening over a period of time that are incorporated in the training model, which we pursue in the future.

Acknowledgement. This research was supported by Grobal Research Laboratory (GRL) Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2016K1A1A2912757). D. M. and D. N. conceived the idea

and wrote the current version of the paper. M. O. did the experiments and contributed to an initial draft of the paper. S. C. contributed to reproducing and examining the results for consistency, and helping with the final writing of the paper. M. O. and S. C. contributed equally to this work.

REFERENCES

[1] Sultan Alshamrani, Mohammed Abuhamad, Ahmed Abusnaina, and David Mohaisen. Investigating online toxicity in users interactions with the mainstream media channels on youtube. In Stefan Conrad and Ilaria Tiddi, editors, *Proceedings of the CIKM 2020 Workshops co-located with 29th ACM International Conference on Information and Knowledge Management (CIKM 2020), Galway, Ireland, October 19-23, 2020*, volume 2699 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020.

[2] Sultan Alshamrani, Ahmed Abusnaina, Mohammed Abuhamad, Anho Lee, Dae-Hun Nyang, and David Mohaisen. An analysis of users engagement on twitter during the COVID-19 pandemic: Topical trends and sentiments. In Sriram Chellappan, Kim-Kwang Raymond Choo, and NhatHai Phan, editors, *Computational Data and Social Networks - 9th International Conference, CSoNet 2020, Dallas, TX, USA, December 11-13, 2020, Proceedings*, volume 12575 of *Lecture Notes in Computer Science*, pages 73–86. Springer, 2020.

[3] Sultan Alshamrani, Ahmed Abusnaina, Mohammed Abuhamad, Daehun Nyang, and David Mohaisen. Hate, obscenity, and insults: Measuring the exposure of children to inappropriate comments in youtube. In Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia, editors, *Companion of The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 508–515. ACM / IW3C2, 2021.

[4] Sultan Alshamrani, Ahmed Abusnaina, and David Mohaisen. Hiding in plain sight: A measurement and analysis of kids' exposure to malicious urls on youtube. In *5th IEEE/ACM Symposium on Edge Computing, SEC 2020, San Jose, CA, USA, November 12-14, 2020*, pages 321–326. IEEE, 2020.

[5] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*, 2018.

[6] Samuel Barham and Soheil Feizi. Interpretable adversarial training for text. *arXiv preprint arXiv:1905.12864*, 2019.

[7] Alessio Bechini, Alessandro Bondielli, Pietro Ducange, Francesco Marcelloni, and Alessandro Renda. Addressing event-driven concept drift in twitter stream: a stance detection application. *IEEE Access*, 2021.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[9] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *arXiv preprint arXiv:1909.03683*, 2019.

[10] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[11] Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. Towards robustness against natural language word substitutions. In *International Conference on Learning Representations*, 2020.

[12] Cicero Dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

[13] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[14] Zhitao Gong, Wenlu Wang, Bo Li, Dawn Song, and Wei-Shinn Ku. Adversarial texts with gradient methods. *arXiv preprint arXiv:1801.07175*, 2018.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[16] Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. Cnn-based chinese ner with lexicon rethinking. In *IJCAI*, pages 4982–4988, 2019.

[17] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.

[18] Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*, 2018.

[19] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[20] Bartosz Kurlej and Michal Wozniak. Active learning approach to concept drift problem. *Logic Journal of IGPL*, 20(3):550–559, 2012.

[21] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[22] Hong Liu, Rongrong Ji, Jie Li, Baochang Zhang, Yue Gao, Yongjian Wu, and Feiyue Huang. Universal adversarial perturbation via prior driven uncertainty approximation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2941–2949, 2019.

[23] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[25] John X Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. Reevaluating adversarial examples in natural language. *arXiv preprint arXiv:2004.14174*, 2020.

[26] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020.

[27] Marwan Omar, Soohyeon Choi, DaeHun Nyang, and David Mohaisen. Robust natural language processing: Recent advances, challenges, and future directions. *arXiv preprint arXiv:2201.00768*, 2022.

[28] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198, 2012.

[29] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[30] Tianlu Wang, Xuezhi Wang, Yao Qin, Ben Packer, Kang Li, Jilin Chen, Alex Beutel, and Ed Chi. Cat-gen: Improving robustness in nlp models via controlled adversarial text generation. *arXiv preprint arXiv:2010.02338*, 2020.

[31] Catherine Wong. Dancin seq2seq: Fooling text classifiers with adversarial text example generation. *arXiv preprint arXiv:1712.05419*, 2017.

[32] Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-wei Chang, and Xuanjing Huang. Defense against adversarial attacks in nlp via dirichlet neighborhood ensemble. *arXiv preprint arXiv:2006.11627*, 2020.

[33] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. *arXiv preprint arXiv:1909.11764*, 2019.

[34] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. *Big data analysis: new algorithms for a new society*, pages 91–114, 2016.