Backdoor Attacks Against Text Classification Models: A Comprehensive Benchmark and Ensemble-Based Defenses

by

Egehan Eralp

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

April 21, 2025

Backdoor Attacks Against Text Classification Models: A Comprehensive Benchmark and Ensemble-Based Defenses

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Egehan Eralp

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Asst. Prof. Mehmet Emre Gürsoy (Advisor)

_____

Prof. Alptekin Küpçü

_____

Prof. Yücel Saygın

Date: _____

ABSTRACT

Backdoor Attacks Against Text Classification Models: A Comprehensive Benchmark and Ensemble-Based Defenses
Egehan Eralp
Master of Science in Computer Science and Engineering
April 21, 2025

Textual backdoor attacks present a critical threat to the security and trustworthiness of NLP systems by embedding stealthy triggers in training data. Such attacks enable adversaries to manipulate model predictions without harming performance on clean inputs. This thesis provides a comprehensive benchmark study of backdoor attacks targeting text classification models, encompassing both traditional machine learning models (Logistic Regression, Naive Bayes, Decision Tree, Random Forest) and neural architectures (LSTM, DistilBERT, BERT, RoBERTa). We evaluate various attack methods—including AddSent, WordInj, SynBkd, StyleBkd, and BITE—across multiple benchmark datasets (IMDb, SST-2, HateSpeech, Tweet) under varying poison rates (0.5% to 10%).

Empirical results reveal that transformer-based models, while achieving high clean accuracy, are especially vulnerable to backdoor triggers. AddSent emerges as the most potent attack, consistently achieving the highest attack success rates (ASRs). Style-transfer attacks (e.g., using Bible or Shakespeare styles) also remain highly effective, often reaching ASRs above 98% at just 3% poison rate. Increasing poison rates amplify both ASR and detectability, posing trade-offs for adversaries and defenders. Traditional models suffer greater drops in clean accuracy at higher poison rates, while transformer models often preserve clean accuracy despite hidden triggers—making them harder to detect in real-world settings.

To counter these attacks, we propose ensemble-based defenses that combine multiple model architectures at inference time. Through experimental evaluations, we observe that ensembles of traditional models successfully reduce ASR but also compromise accuracy. Ensembles of transformer models achieve high accuracy, but do not succeed in reducing ASR to acceptable levels. We therefore propose joint ensembles that combine traditional models with modern transformer models. Our

approach significantly reduces ASR while maintaining competitive clean accuracy, highlighting the benefits of architectural diversity and serving as a potential defense mechanism for building backdoor-resilient text classification systems.

ÖZETÇE

Metin Sınıflandırma Modellerine Yönelik Arka Kapı Saldırıları: Kapsamlı Bir Karşılaştırma ve Topluluk Tabanlı Savunmalar

Egehan Eralp

Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans

21 Nisan 2025

Metinsel arka kapı saldırıları, eğitim verilerine gizli tetikleyiciler yerleştirerek NLP sistemlerinin güvenliği ve güvenilirliği için kritik bir tehdit oluşturmaktadır. Bu tür saldırılar, saldırganların temiz girdiler üzerindeki performansı etkilemeden model tahminlerini manipüle etmesine olanak tanımaktadır. Bu tez, metin sınıflandırma modellerini hedef alan arka kapı saldırılarına yönelik kapsamlı bir karşılaştırmalı çalışma sunmaktadır; bu çalışma, geleneksel makine öğrenimi modellerini (Lojistik Regresyon, Naive Bayes, Karar Ağacı, Rastgele Orman) ve sinir ağı mimarilerini (LSTM, DistilBERT, BERT, RoBERTa) kapsamaktadır. Çalışmamızda, çeşitli saldırı yöntemlerini — AddSent, WordInj, SynBkd, StyleBkd ve BITE — farklı zehirleme oranları (%0.5 ile %10 arası) altında, çeşitli karşılaştırmalı veri setleri (IMDb, SST-2, HateSpeech, Tweet) üzerinde değerlendirmekteyiz.

Ampirik sonuçlarımız, temiz doğruluk oranları yüksek olan transformer tabanlı modellerin, arka kapı tetikleyicilerine karşı özellikle savunmasız olduğunu ortaya koymaktadır. AddSent yöntemi, tutarlı bir şekilde en yüksek saldırı başarı oranlarına (ASR) ulaşarak en güçlü saldırı olarak öne çıkmaktadır. Stil transferi tabanlı saldırılar (örneğin, İncil veya Shakespeare tarzlarının kullanıldığı saldırılar) da oldukça etkili kalmakta ve yalnızca %3 zehirleme oranı ile %98'in üzerinde ASR elde edebilmektedir. Zehirleme oranlarının artırılması hem ASR'yi hem de saldırının tespit edilebilirliğini artırmakta; bu durum, saldırganlar ve savunmacılar açısından önemli bir denge problemi oluşturmaktadır. Geleneksel modeller, yüksek zehirleme oranlarında temiz doğrulukta daha fazla düşüş yaşarken; transformer modeller, gizli tetikleyicilere rağmen genellikle temiz doğruluğu koruyabilmekte — bu da onları gerçek dünya senaryolarında tespit edilmesi daha zor hale getirmektedir.

Bu saldırılara karşı koymak için, tahmin (inference) aşamasında birden fazla model mimarisini birleştiren topluluk (ensemble) tabanlı savunmalar önermekteyiz.

Deneysel değerlendirmelerimiz sonucunda, geleneksel modellerden oluşan toplulukların ASR'yi başarılı bir şekilde azaltabildiğini ancak aynı zamanda doğrulukta kayıplara yol açtığını gözlemliyoruz. Transformer modellerden oluşan topluluklar ise yüksek doğruluk elde ederken, ASR'yi kabul edilebilir seviyelere düşürmekte başarısız kalmaktadır. Bu nedenle, geleneksel modeller ile modern transformer modelleri bir araya getiren ortak (joint) topluluklar önermekteyiz. Önerdiğimiz yaklaşım, rekabetçi temiz doğruluğu korurken ASR'yi önemli ölçüde azaltmaktadır; bu da mimari çeşitliliğin faydalarını göstermekte ve arka kapı saldırılarına karşı dayanıklı metin sınıflandırma sistemleri geliştirmek için potansiyel bir savunma mekanizması sunmaktadır.

ACKNOWLEDGMENTS

I would like to begin by expressing my heartfelt appreciation to Asst. Prof. Dr. Mehmet Emre Gürsoy for his exceptional mentorship, insightful guidance, and continuous support throughout my Master's program. His dedication not only helped shape my academic growth but also inspired me to contribute by mentoring undergraduate students.

I would like to thank Prof. Dr. Alptekin Küpçü from Koç University and Prof. Dr. Yücel Saygın from Sabancı University for being part of my thesis jury committee, and for their invaluable comments and suggestions to improve this work.

I would like to express my deepest thanks to my wife, Minel Enginyurt Eralp, whose unwavering belief in me, patience, and encouragement made all the difference throughout this journey. I extend my heartfelt gratitude to my mother Sema Eralp, my father Cengiz Eralp, and my sisters Eylül Eralp and Öykü Eralp for their love and support throughout my life. I thank all members of the SPADE Lab for providing a productive and collaborative research environment, especially Aslı Atabek for her enduring friendship and support during my Master's studies.

Lastly, this research was supported by The Scientific and Technological Research Council of Turkiye (TUBITAK) under project number 125E059. I thank TUBITAK for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

NLP	Natural Language Processing
ML	Machine Learning
DL	Deep Learning
LR	Logistic Regression
NB	Naive Bayes
DT	Decision Tree
RF	Random Forest
RNN	Recurrent Neural Network
DNN	Deep Neural Network
LSTM	Long Short-Term Memory
CA	Classification Accuracy
ASR	Attack Success Rate
PV	Paragraph Vector
DM	Distributed Memory
DBOW	Distributed Bag of Words

Chapter 1

INTRODUCTION

The field of natural language processing (NLP) has seen remarkable advancements in recent years, driven by developments in deep learning and language models. NLP now supports a wide array of applications such as sentiment analysis, spam detection, content moderation, question answering, and text generation. In particular, **text classification** remains a fundamental task in NLP, where the goal is to assign predefined categories (class labels) to textual samples. Several real-world applications of NLP rely on text classification, e.g., sentiment analysis, spam detection, hate speech detection, and document type classification.

On the other hand, as NLP models become more integrated into critical systems, concerns about their security vulnerabilities have also grown. Among these threats, **backdoor attacks** pose a particularly insidious risk. In a backdoor attack, the attacker poisons a small subset of the training data (or directly manipulates model weights) so that the model learns to associate an attacker-chosen trigger pattern with an attacker-chosen class label. Importantly, the model continues to perform normally on samples that do not contain the trigger, maintaining high accuracy on clean test data. However, when the trigger (e.g., a specific word or text style) appears in an input, the backdoored model's prediction is immediately biased towards the attacker-chosen label. This dual behavior allows backdoored models to evade detection during testing while still harboring a serious vulnerability.

Backdoor attacks are especially feasible in today's systems when the training pipeline involves untrusted data or third-party models. For example, an attacker can inject poisoned samples into a publicly released dataset or publish a language model involving a backdoor (e.g., on Huggingface). An unsuspecting practitioner

can incorporate this data or model into their pipeline and obtain a backdoored text classifier. When the backdoored classifier is deployed, it can cause misclassifications of hate speech, toxic content, or spam e-mails.

Motivated by the plausibility and threat of backdoor attacks, this thesis provides a comprehensive benchmark study on backdoor attacks against text classification models, investigating the impacts of different attack strategies, model architectures, poison rates, and other relevant aspects. We evaluate five popular attack strategies (AddSent, WordInj, SynBkd, StyleBkd, and BITE) across a diverse set of models, including traditional machine learning models (Doc2Vec + Logistic Regression, Naive Bayes, Decision Tree, Random Forest), sequence-based deep learning models (LSTM), and transformer-based models (DistilBERT, BERT, RoBERTa). Our benchmark spans four representative datasets (IMDb, SST-2, HateSpeech, and Tweet) and systematically analyzes the relationship between backdoor attack success rate (ASR), benign classification accuracy (CA), model complexity, and poison rate. In particular, eight research questions (RQ1-RQ8) are formulated to guide the experimental analysis, spanning issues related to the impacts of model type on CA and ASR, comparative analysis of different backdoor attacks, impacts of varying poison rate and model complexity on CA and ASR, and trade-off visualizations between CA and ASR. Furthermore, ablation studies are performed to analyze the impacts of attack-related and model-related parameters.

To mitigate backdoor attacks, we propose **ensemble-based defenses** that aggregate predictions from multiple models at inference time. Unlike existing defenses that rely on identifying or filtering out poisoned inputs, our approach is agnostic to the attack type and does not require access to the poisoned training data. We show that ensembles of traditional models offer robustness at the expense of accuracy, and transformer-only ensembles offer high accuracy but limited robustness. We therefore recommend the use of a *joint ensemble*, which combines both traditional models and transformer models. We show that the joint ensemble strikes a favorable balance between model accuracy and robustness (CA and ASR). In particular, it significantly reduces ASR across multiple attack scenarios while maintaining competitive CA, demonstrating the potential of architectural diversity as a defense strategy.

1.1 Thesis Contributions

In short, this thesis studies backdoor attacks in text classification models and makes the following main contributions:

- **Comprehensive benchmark of textual backdoor attacks:** We construct an extensive benchmark setup to evaluate the impact of backdoor attacks on various text classification models, including traditional machine learning models, neural networks, and transformer-based architectures. Our benchmark includes five attack types, four datasets, varying poison rates (ranging from 0.5% to 10%), and other relevant parameters.

- **Experimental analysis of model vulnerability and attack impacts:** We use classification accuracy (CA) and attack success rate (ASR) as key evaluation metrics. We systematically construct eight research questions to analyze the impacts of model type and complexity, poison rate, and attack type on the CA-ASR trade-offs. Our findings reveal that transformer-based models, while highly effective for text classification, exhibit greater vulnerability to backdoor triggers. Additionally, the AddSent attack achieves the highest ASR values, highlighting the need for more effective defense mechanisms.

- **Ensemble-based defenses:** We propose an ensemble-based inference-time defense strategy that leverages both traditional and transformer-based models to mitigate the effects of backdoor attacks. Unlike existing defenses that focus on detecting and removing specific backdoor triggers from training datasets or filtering test samples, our approach is designed to be attack- and dataset-agnostic. By integrating diverse models within an ensemble, we enhance robustness without significantly compromising CA. Our results show that the proposed joint ensemble method effectively balances CA and ASR across different attacks. For example, in the AddSent attack conducted on the IMDb dataset with a 1% poison rate, replacing the BERT model with our joint ensemble solution resulted in a 3.26% decrease in CA, while achieving a 63% reduction in ASR.

1.2 Thesis Organization

The remainder of this thesis is structured as follows. Chapter 2 provides an overview of prior research on textual backdoor attacks and existing defense mechanisms, highlighting their strengths and limitations. Chapter 3 describes our comprehensive benchmark setup, explaining text classification models, backdoor attack strategies, datasets, and metrics used in our study. Chapter 4 presents a detailed analysis of our experimental results, discussing the impact of different attack methods, model architectures, and poison rates guided by research questions (RQs). Chapter 5 introduces our proposed ensemble-based defense mechanism, explaining its design and evaluating its effectiveness against various backdoor attacks. Chapter 6 summarizes our key findings, discusses the implications of our work, and concludes this thesis.

Chapter 2

RELATED WORK

Backdoor attacks were initially applied to computer vision models, leading to a number of academic studies that paved the way for our work. For example, [Gu et al., 2017] compromise an image classifier by embedding a square-shaped trigger pattern into the training images, which causes the model to produce incorrect classifications for test images containing the same trigger at inference time. [Liu et al., 2018] introduced trojaning attacks on image-based neural networks, embedding malicious behaviors in tasks like face recognition and autonomous driving, while maintaining high accuracy on benign inputs. The successful outcomes produced by backdoor attacks in the field of computer vision, coupled with the significant momentum gained by the transformer architecture proposed by [Vaswani, 2017], have led to increased interest in backdoor attacks and defenses within the field of NLP and text classification.

2.1 Backdoor Attacks in Text Classification

Backdoor attacks in the field of text classification can be examined under two main categories: attacks based on data manipulation and model manipulation.

In **data manipulation-based backdoor attacks**, the trigger pattern is injected into training samples, and the model learns from this poisoned data. The model learning algorithm itself is not manipulated. Attacks in this category include the insertion of fixed characters, words, sentences [Chen et al., 2021], or modifications to syntactic structures or stylistic features within the training data. [Dai et al., 2019] proposed the first backdoor attack to text classification by targeting LSTM-based sentiment analysis. Their method performs the attack by inserting emotionally neutral sentences into samples. Experiments on sentiment analysis with

IMDb reviews [Maas et al., 2011] demonstrated that even with a poisoning rate as low as 1%, the attack achieved a success rate of 95%, highlighting the vulnerability of LSTM and RNN-based architectures to backdoor threats. Subsequently, [Kurita et al., 2020] proposed an attack by inserting a fixed word into the dataset and carried out experiments on pre-trained models with a transformer architecture. Their attack method utilizes short trigger keywords such as "cf", "mn" and "tq". In contrast to fixed-word approaches, [Chan et al., 2020] employed a conditional adversarially regularized autoencoder (CARA) to generate backdoored training samples. This method does not rely on fixed phrases as triggers but instead uses an intermediate model to create poisoned samples that appear more natural to human readers, despite their semantics differing significantly from the original text. This approach enhances stealthiness, making detection more challenging.

Instead of focusing on word or phrase injection, [Qi et al., 2021b] and [Qi et al., 2021c] utilized text style and syntactic structure as backdoor triggers. Their work highlights the significant security risks posed by task-irrelevant features, such as stylistic or syntactic elements of text, which are often overlooked in robustness evaluations. These subtle manipulations make the attacks harder to detect and demonstrate vulnerabilities beyond lexical-level modification. Building upon these approaches that emphasize stealth and naturalness, [Yan et al., 2023] proposed BITE, a textual backdoor attack that poisons training data by iteratively injecting trigger words. Unlike traditional fixed-word triggers, BITE establishes strong correlations with the target label through word-level perturbations, enhancing its ability to remain undetected.

In **model manipulation-based backdoor attacks**, the attacker focuses on directly altering a model's internal parameters, architecture, or training process to embed triggers. Attacks in this category can be further divided into two subcategories: output-based attacks, which have access to the forward-pass process, and gradient-based attacks, which have access to the backward gradient update process. Among output-based attacks is [Shen et al., 2021], who explored backdoor vulnerabilities in pre-trained NLP models by mapping inputs with triggers to predefined output representations (POR), allowing the backdoor to persist across various down-

stream tasks without prior knowledge of specific labels. Building on this concept, [Li et al., 2021] proposed a Layerwise Weight Poisoning (LWP) attack that targets the lower layers of pre-trained language models. By embedding triggers in the early layers, their method ensures that the backdoor survives fine-tuning, as higher layers are more affected during model adaptation. Further extending these ideas, [Zhang et al., 2023] proposed a neuron-level backdoor attack (NeuBA) that manipulates output neuron representations in pre-trained models, enabling malicious behavior across various downstream tasks without prior task knowledge. Their experiments in both NLP and computer vision demonstrate that backdoored models retain their functionality after fine-tuning, highlighting a universal vulnerability.

In gradient-based attacks, the adversary exploits access to the model's gradient updates to introduce malicious behaviors at the parameter or embedding level. The first work to explore gradient-based model manipulation in text classification is by [Kurita et al., 2020], who conducted backdoor experiments on pre-trained transformer models by inserting fixed trigger words into the dataset while simultaneously modifying the embedding layer to reinforce the backdoor. Building on this idea, [Qi et al., 2021d] introduced Learnable Word Substitution (LWS), a more advanced gradient-based attack that subtly modifies model parameters through synonym-based word substitutions. This method maintains the original syntax and semantics of the text while embedding the backdoor in the model's learned representations, making detection significantly more challenging. [Yang et al., 2021c] further refined gradient-based attacks by addressing the stealth limitations of earlier methods. They proposed the SOS (Stealthy Backdoor Attack with Stable Activation) framework, which employs negative data augmentation alongside word embedding modifications to craft more covert and stable backdoor triggers. By stabilizing neuron activations during training, SOS ensures that the backdoor remains effective. [Yang et al., 2021a] introduced a data-free backdoor attack that manipulates a single word embedding vector to induce misclassifications. This approach eliminates the need for task-specific datasets by leveraging general text corpora.

Recent research has introduced comprehensive surveys and frameworks for understanding backdoor attacks, not only in text classification but also in other domains.

We refer the interested reader to these resources [Cui et al., 2022, Sheng et al., 2022]. As opposed to many prior works that verbally survey backdoor attacks, this thesis provides a quantitative benchmark and analysis.

2.2 Defenses Against Backdoor Attacks

The growing sophistication and variety of backdoor attacks highlight the pressing need for defense mechanisms. We present defenses under primarily two categories: inference-time defenses, which aim to detect and block backdoor triggers during model deployment, and training-time defenses, which focus on identifying and mitigating poisoned data or malicious model modifications during the training phase.

Inference-time defenses aim to identify test inputs that contain potential triggers, preventing backdoor-activated misclassifications during model deployment. These methods typically analyze incoming test samples using techniques like input transformation, anomaly detection, or activation pattern analysis to detect abnormal behaviors associated with backdoor triggers.

A notable inference-time defense is ONION, proposed by [Qi et al., 2021a], which relies on outlier word detection to identify potential backdoor triggers. ONION removes suspicious words that deviate from the expected distribution within the input and evaluates the model's prediction stability. This approach has proven effective against various backdoor attacks on BiLSTM and BERT models. [Gao et al., 2022] introduced STRIP-ViTA, a multi-domain backdoor detection method that operates across vision, text, and audio tasks, demonstrating strong performance in detecting backdoor attacks with minimal false acceptance and rejection rates. [Yang et al., 2021b] proposed a method based on robustness-aware perturbations (RAP). Their approach utilizes the robustness differences between clean and poisoned samples to detect backdoor triggers by introducing a specific perturbation and observing the model's response. STRIP and RAP detect malicious test samples by analyzing the model's prediction stability when words in the input are perturbed. Test samples identified as poisoned through this process are subsequently discarded.

The defenses we propose in Chapter 5 fall under the category of inference-time

defenses. Unlike existing inference-time defenses that focus on detecting and filtering poisoned inputs, our approach takes a fundamentally different strategy by mitigating the impact of backdoor triggers rather than explicitly identifying them. ONION [Qi et al., 2021a], STRIP-ViTA [Gao et al., 2022], and RAP [Yang et al., 2021b] rely on perturbation-based detection or outlier analysis to identify anomalous behavior. While effective, these methods often require predefined heuristics for anomaly detection, making them susceptible to adaptive attacks where triggers are crafted to evade detection. In contrast, our proposed defense leverages the diversity of multiple models, including both traditional and transformer-based architectures, to dilute the impact of backdoor triggers without necessitating explicit detection mechanisms. By integrating models with varying decision boundaries and feature extraction strategies, our approach enhances robustness against backdoor attacks while maintaining competitive classification accuracy.

Training-time defenses aim to sanitize the poisoned training dataset to avoid the backdoor from being learned. These defenses typically analyze training data for anomalies, such as trigger patterns or abnormal distributions, and apply techniques like data filtering or adversarial training to remove or neutralize poisoned samples.

[Chen and Dai, 2021] introduced Backdoor Keyword Identification (BKI), which identifies backdoor keywords by analyzing the changes in LSTM hidden states and removes associated training samples. Experiments across various datasets demonstrated the method's effectiveness in mitigating backdoor attacks without relying on trusted datasets. [Cui et al., 2022] proposed clustering-based poisoned sample filtering for backdoor-free training (CUBE). By analyzing representations in the latent space, CUBE effectively distinguishes malicious samples from benign ones, providing a simple yet robust baseline for defending against textual backdoors. [Shao et al., 2021] proposed a backdoor defense model based on detection and reconstruction (BDDR), which detects suspicious words and reconstructs text through deletion or replacement using a masked language model such as BERT. Their experiments demonstrated notable reductions in attack success rates for both word-level and sentence-level backdoor attacks without significantly impacting model accuracy.

[Fan et al., 2021] proposed InterRNN, which focuses on backdoor detection in

RNN-based text classification systems from an interpretability perspective. Their method constructs a nondeterministic finite automaton (NFA) to abstract the RNN model and detect suspicious triggers by analyzing the differences in word-level behavior between normal and backdoor-embedded sentences. Suspicious samples containing these triggers are subsequently removed from the training set. Additionally, when model retraining is not possible, InterRNN can be applied at inference time by removing the detected trigger words from test inputs to prevent backdoor activation. [Azizi et al., 2021] proposed T-Miner, a defense for detecting backdoor attacks in DNN-based text classification. Their approach utilizes a sequence-to-sequence generative model to probe the classifier and identify potential trigger phrases by analyzing perturbations that cause misclassifications. Once a backdoored model is detected, it can either be discarded or patched to remove the trigger effect before deployment. [Yan et al., 2023] proposed DeBITE to mitigate backdoor triggers, which are based on trigger words that create strong label correlations. DeBITE works by identifying and removing words with anomalous label distributions, preventing the model from learning these malicious patterns.

Chapter 3

BENCHMARK SETUP

3.1 Preliminaries and Background

Text Classification. In a typical text classification setup, a dataset of training samples $\mathcal{D}_{train} = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ is used to train a classification model $\mathcal{M}$, where x_i denotes the text document and y_i denotes the class label. For example, x_i can be an e-mail or movie review and y_i can be either 0 or 1, corresponding to positive or negative sentiment. Given $\mathcal{D}_{train}$, the goal of text classification is to build a model $\mathcal{M}$ that can correctly predict the labels of previously unseen test samples. That is, for each test sample $(x_t, y_t) \in \mathcal{D}_{test}$, denoting the prediction of $\mathcal{M}$ by $\mathcal{M}(x_t) \rightarrow y_t^*$, it is desired that $y_t = y_t^*$. The *classification accuracy (CA)* of $\mathcal{M}$ can be measured as:

$$\text{CA} = \frac{\#\text{ of samples } (x_t, y_t) \in \mathcal{D}_{test} \quad \text{s.t.} \quad \mathcal{M}(x_t) \rightarrow y_t^* \text{ and } y_t^* = y_t}{|\mathcal{D}_{test}|} \tag{3.1}$$

CA represents the performance of a trained text classification model on benign test data. It measures the model's effectiveness in performing its downstream task (text classification) without any backdoor influence. A higher CA value indicates that the model can perform well on its intended classification task.

Backdoor attacks in text classification. In normal circumstances, $\mathcal{D}_{train}$ consists of benign training samples. However, in an adversarial environment, it is assumed that a portion of $\mathcal{D}_{train}$ is manipulated by the attacker to execute a backdoor attack. The percentage of $\mathcal{D}_{train}$ that is manipulated by the attacker is called the *poison rate*. In a backdoor attack, the attacker manipulates the samples by injecting a *trigger* pattern. The attacker's goal is to cause $\mathcal{M}$ to associate the trigger pattern with a particular (attacker-chosen) class label $\hat{y}$, such that at test time, when a test sample containing the trigger pattern is presented to the backdoored $\mathcal{M}$, $\mathcal{M}$ will predict that sample as belonging to class $\hat{y}$.

The *Attack Success Rate (ASR)* of a backdoor attack can be measured as follows. Let $\mathcal{D}_{bd} \subseteq \mathcal{D}_{test}$ denote a subset of the test dataset, and let $g(\cdot)$ denote the function that injects the trigger to a sample. Then:

$$\text{ASR} = \frac{1}{\mathcal{D}_{bd}} \sum_{i=1}^{|\mathcal{D}_{bd}|} \mathbb{1}\big[\mathcal{M}(g(x_i)) = \hat{y}\big] \tag{3.2}$$

where $\mathbb{1}[\cdot]$ is the indicator function. Intuitively, ASR measures: "What fraction of originally clean test samples are classified as the attacker-chosen $\hat{y}$ by the backdoored model after the backdoor trigger is injected to them?"

In a typical backdoor attack, the attacker aims to increase ASR while simultaneously maintaining high CA. ASR is between 0 and 1 by definition, and an ASR close to 1 indicates that the attack is more effective. Maintaining high CA, e.g., CA close to a non-backdoored model, is important for stealth. If the attack has near-zero or negligible impact on CA, then it is less likely to raise suspicion and can remain stealthy.

3.2 Datasets and Classification Tasks

Our benchmark contains four datasets corresponding to four text classification tasks, including sentiment analysis, hate speech detection, and emotion recognition, with different numbers of classes. The datasets are summarized in Table 3.1. The table shows the number of samples in the training, validation, and test datasets, the number of classes, and the average lengths of the text samples in terms of word count. To increase the breadth of our benchmark, we included both binary and multi-class datasets, as well as datasets with short and long samples.

IMDb [Maas et al., 2011] is a movie review dataset, containing 25000 highly polar movie reviews for training and 25000 for testing. It is commonly used for binary sentiment classification. In our experiments, we chose $\hat{y} =$ **positive** sentiment (1) as the target label. Thus, the goal of the backdoor attack is for the backdoored model to predict the positive label when the corresponding backdoor trigger is activated.

SST-2 [Socher et al., 2013] is another binary sentiment classification dataset containing movie reviews, but the average sample length in SST-2 is much smaller

Table 3.1: Statistics of the datasets.

Dataset	$\lvert\mathcal{D}_{train}\rvert$	$\lvert\mathcal{D}_{val}\rvert$	$\lvert\mathcal{D}_{test}\rvert$	# of Classes	Avg. Sample Length
IMDb	22500	2500	25000	2	231.2
SST-2	6920	872	1821	2	19.3
HateSpeech	7703	1000	2000	2	18.1
Tweet	3257	374	1421	4	16.1

than IMDb. The class distribution in $\mathcal{D}_{train}$ is 52.2% positive and 47.8% negative labels. Similar to IMDb, we chose $\hat{y} =$ **positive** sentiment as the target label.

HateSpeech is a binary hate speech detection dataset containing forum posts. The class distribution is relatively unbalanced, i.e., 88.8% no hate and 11.2% hate labels. In our experiments, $\hat{y} =$ **no hate** was selected as the target label. Thus, the goal of the backdoor attack is for the backdoored model to predict "no hate" for test samples containing the trigger pattern, even if they contain hate speech.

Tweet (TweetEval-Emotion) [Mohammad et al., 2018] is a multi-class emotion recognition dataset (as opposed to binary classification in the other datasets). There are four classes: joy, optimism, sadness, and anger. We selected $\hat{y} =$ **anger** as the target label for the backdoor attacks.

3.3 Model Types

The text classification model $\mathcal{M}$ can be of several types. Classical text classification methods begin with a form of feature extraction (e.g., tf-idf, bag of words, or word embeddings such as Doc2Vec [Le and Mikolov, 2014]) followed by a traditional machine learning algorithm such as Logistic Regression, Decision Tree, or SVM. With the rise of deep learning, classical methods were replaced by the likes of Convolutional Neural Networks (e.g., TextCNN), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM). The aforementioned model types are typically trained from scratch using $\mathcal{D}_{train}$. In contrast, transformer-based models have recently been providing state-of-the-art accuracy [Devlin et al., 2018]; therefore, they

have become popular choices in text classification. Typically, pre-trained BERT, RoBERTa, or DistilBERT models are obtained (e.g., from Huggingface) and then fine-tuned using $\mathcal{D}_{train}$.

In order to analyze the impact of model complexity on backdoor attacks, as well as to demonstrate attack effectiveness on a large variety of models, we used models from three categories in our benchmark: (i) traditional Doc2Vec-based models, (ii) LSTM models, and (iii) transformer-based models: DistilBERT, BERT, and RoBERTa. Below, we describe models from each of these three categories.

3.3.1 Traditional Doc2Vec-Based Models

Machine learning algorithms, particularly those relying on numerical computations, need their inputs to be in a numerical format. Since raw text data is naturally non-numerical and unstructured, it must be converted into a structured numerical form. To achieve this, techniques like word embeddings (e.g., Word2Vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014]), document embeddings (e.g., Doc2Vec [Le and Mikolov, 2014]), and contextual embeddings (e.g., BERT [Devlin et al., 2018], GPT2 [Radford et al., 2019]) can be employed.

In our study, we utilized document embeddings. Document embedding involves converting an entire text document x_i, which could be a paragraph, article, or book, into a single vector. This method captures the meaning and context of individual sentences as well as the relationships and coherence between sentences throughout the document. We utilized Doc2Vec [Le and Mikolov, 2014], which is a widely used technique in NLP, for document embedding. Doc2Vec was developed as an extension of Word2Vec [Mikolov et al., 2013], which focuses on representing words as numerical vectors. While Word2Vec learns word embeddings, Doc2Vec is designed to learn document embeddings. It transforms each document into a fixed-length vector in a high-dimensional space, positioning similar documents close to each other. There are two main variants of Doc2Vec: Distributed Memory (DM) and Distributed Bag of Words (DBOW).

- The principal concept of DM is inspired by the CBOW model from Word2Vec.

In DM, consecutive words are randomly sampled from a paragraph to predict a central word using the context words and a paragraph ID as input. The context words help predict the target word, while the document ID captures the document's overall meaning.

- DBOW is a simpler variant of Doc2Vec that emphasizes understanding the distribution of words in a text rather than their meaning. It is better for analyzing text structure rather than content. In DBOW, each document in the corpus is assigned a unique vector representation without separate word vectors. The algorithm uses the document vector to predict the probability of each word in the document. Context words are ignored in the input, but the model is trained to predict words randomly sampled from the paragraph in the output.

In summary, DM accounts for both word order and document context, making it more effective for capturing the semantic meaning of documents. In contrast, DBOW is simpler and faster to train, focusing on capturing the distributional properties of words in a corpus. We use both DM and DBOW in our benchmark, and we compare them in terms of CA and ASR.

When training Doc2Vec, the negative sampling, hierarchical softmax, and min-count (ignoring words based on minimum frequency) parameters were selected uniformly across all datasets. For other hyperparameters of Doc2Vec, the grid search technique was used on each individual dataset to identify the optimal values. The optimal Doc2Vec hyperparameter values found for each dataset are shown in Table 3.2. All other hyperparameters were used with their default settings.

After converting raw text data into numerical vectors using Doc2Vec, we trained four machine learning models: Logistic Regression (LR), Gaussian Naive Bayes (NB), Decision Tree (DT), and Random Forest (RF).

- For LR, we employed the LR model from the scikit-learn library. The model was configured with the L2 (Ridge) regularizaton penalty, a regularization strength (C) of 1.0, and the LBFGS solver, which is well-suited for small to

Table 3.2: Hyperparameters of Doc2Vec model on datasets.

Parameter \ Dataset	IMDb	SST-2	HateSpeech	Tweet
DM/DBOW	DBOW	DM	DM	DM
Vector Size	50	100	50	100
Window	5	6	3	2
Negative Sampling	5	5	5	5
Hierarchical Softmax	0	0	0	0
Min. Count	2	2	2	2

medium-sized datasets. Additionally, the model used an automatic selection of multi-class strategies based on the dataset characteristics, defaulting to the OVR (one-vs-rest) scheme. The maximum number of iterations was set to 100, and the tolerance for stopping criteria was 1e-4. All other hyperparameters were left at their default values.

- In NB, the model assumes a Gaussian distribution for the input features and estimates class-conditional probabilities accordingly. The default settings include no prior probabilities, allowing the model to learn class priors directly from the training data, and the default smoothing parameter is set to 1e-9, which helps to stabilize calculations by adding a small value to the variance to prevent division by zero. All other hyperparameters were left at their default values.

- In DT, we utilized the Gini Impurity as the split criterion. Instead of employing a random split strategy, we applied the best split strategy. We did not impose a maximum depth constraint on the DT, allowing the nodes to expand until all leaves were pure or until all leaves contained fewer samples than the "minimum samples to split" threshold. We set the minimum number of samples required to split an internal node to 2.

- Random Forest (RF) is an ensemble learning method that utilizes multiple decision trees. In our RF implementation, we employed the same split criteria and thresholds as the DT model. An ensemble was formed by selecting 100 trees within the RF. The bootstrap method was enabled to allow the model to train on different subsets of the data, which enhances the model's generalization ability and reduces the problem of overfitting.

3.3.2 LSTM Models

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) designed to effectively learn and capture long-term dependencies in sequential data. Unlike standard RNNs, LSTMs address the problem of vanishing gradients by incorporating memory cells and gating mechanisms that regulate the flow of information. These gates control what information is added to or removed from the cell state, enabling LSTMs to retain important information over long sequences while discarding irrelevant data. This makes LSTMs particularly powerful for tasks involving sequential data such as time series forecasting, speech recognition, and NLP.

To use an LSTM as a text classifier, the text data is first tokenized and converted into a sequence of word embeddings that serve as input to the LSTM. The LSTM processes this sequential data one word at a time, maintaining a hidden state that captures the context and dependencies across the sequence. As the LSTM progresses through the text, it updates its cell state and hidden state using the input, forget, and output gates, thereby learning which information to remember and which to discard. After processing the entire text, the hidden state at the final time step encapsulates the learned representation of the text. This final hidden state is then fed into a fully connected layer, followed by a softmax activation function to predict the probability distribution over the possible classes. The class with the highest probability is chosen as the predicted label for the text.

In our work, the architecture of our LSTM solution comprises LSTM layers, feedforward dense layers, and dropout layers. Given that the LSTM solution is employed for a medium-complexity text classification task, we aimed to minimize

Table 3.3: LSTM architecture for IMDb, SST, and HateSpeech datasets

Layer	Type	Output Shape	Activation	Parameter #
1	Embedding	(None, 250, 100)	-	5,000,000
2	Dropout (0.2)	(None, 250, 100)	-	0
3	LSTM	(None, 32)	-	17,024
4	Dense	(None, 256)	ReLU	8,448
5	Dropout (0.2)	(None, 256)	-	0
6	Dense	(None, 1)	Sigmoid	257
Total Trainable Parameter #				5,025,729

the number of neurons in the overall network as much as possible. Consequently, we utilized LSTM models with approximately 5 million parameters. For all datasets, we set the maximum number of words to be used (most frequent words) threshold to 50000, limited the maximum number of words in each input to 250, and chose an embedding dimension of 100 to create our embedding layer. Additionally, for all datasets, we used the AdamW optimizer, set the training epochs to 20, and the batch size to 64 during the training process.

Our LSTM architectures for the binary text classification datasets (IMDb, SST-2, and HateSpeech) are shown in Table 3.3. As shown in the table, the designed LSTM layer contains 32 neurons/hidden states, the output layer utilizes a sigmoid activation function, and cross-entropy loss is used as the loss function. The LSTM architecture for the multi-class dataset Tweet is shown in Table 3.4. For this dataset, the LSTM layer contains 64 neurons/hidden states, the output layer employs a softmax activation function, and to prevent overfitting, the number and rates of dropout layers were increased. Sparse categorical cross-entropy loss was used as the loss function.

Table 3.4: LSTM architecture for Tweet dataset

Layer	Type	Output Shape	Activation	Parameter #
1	Embedding	(None, 250, 100)	-	5,000,000
2	Dropout (0.3)	(None, 250, 100)	-	0
3	LSTM	(None, 64)	-	42,240
4	Dropout (0.2)	(None, 64)	-	0
5	Dense	(None, 256)	ReLU	16,640
6	Dropout (0.3)	(None, 256)	-	0
7	Dense	(None, 4)	Softmax	1028
Total Trainable Parameter #				5,059,908

3.3.3 Transformer-Based Models

We used three transformer-based models in our benchmark: BERT, DistilBERT, and RoBERTa. To adapt the models to classification tasks, we added a classifiation layer at the end of each model and then performed fine-tuning. The summary of the three models is provided in Table 3.5; their details are explained below.

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a pre-trained language model developed by Google [Devlin et al., 2018]. It is designed to understand the context of a word in text by examining the words that come before and after it. BERT uses a transformer architecture, which relies on attention mechanisms to process words in relation to all other words in a sentence rather than sequentially. This allows BERT to capture more nuanced meanings and relationships in language, making it highly effective for a wide range of NLP tasks.

The architecture of BERT consists of several key components. It begins with an embedding layer, which includes word embeddings with a vocabulary size of 30,522 and an embedding dimension of 768. Additionally, it incorporates position embeddings and token type embeddings, both with an embedding dimension of 768. These embeddings are followed by layer normalization and dropout to regularize the training process. The core of BERT's architecture is its encoder, which is composed of 12

Table 3.5: Summary of transformer models

Architecture	Model	Details
DistilBERT	distilbert-base-cased	6-layer, 768-hidden, 12-heads, 65M parameters
BERT	bert-base-cased	12-layer, 768-hidden, 12-heads, 110M parameters
RoBERTa	roberta-base	12-layer, 768-hidden, 12-heads, 125M parameters

identical layers, each called a BertLayer. Each BertLayer includes a self-attention mechanism, implemented through the BertAttention component, which uses linear layers to project the input into queries, keys, and values. These projections allow the model to weigh the relevance of each token to every other token in the sequence. Following the attention mechanism, a BertSelfOutput layer, consisting of a linear transformation, layer normalization, and dropout, processes the attended representations. The intermediate representation is then transformed by a BertIntermediate layer with a linear layer and a GELU activation function, followed by another BertOutput layer that applies another linear transformation, layer normalization, and dropout. BERT also includes a pooler component, which uses a dense layer followed by a Tanh activation function to create a pooled representation of the entire input sequence. This pooled output is particularly useful for classification tasks. Additionally, there is a final dropout layer applied to the pooled output to further regularize the model before it is fed into the classifier.

Using BERT as a text classifier involves adapting its pre-trained language model to a specific text classification task. In our study, we added a classification layer to the end of the pre-trained BERT model (bert-base-cased) and performed **full fine-tuning (FFT)** with 2 training epochs. In full fine-tuning, all the parameters of the pre-trained BERT model, including the transformer layers, are updated during the training process. This contrasts with partial fine-tuning or feature extraction, where some layers are frozen and only a subset of the parameters (often just the classification head) are updated. Full fine-tuning allows the model to adjust all its learned representations to better fit the new task, potentially leading to higher performance. The IMDb, SST-2, and HateSpeech datasets require binary classifica-

tion, so the classification layer at the end of BERT outputs 2 labels. In contrast, the Tweet dataset involves multi-class text classification, necessitating a classification layer that outputs 4 labels. Other than these differences in the classification layer, the underlying BERT model architecture remains consistent.

DistilBERT [Sanh et al., 2019] is a smaller, faster, and lighter (distilled) version of the BERT model. It retains 97% of BERT's language understanding capabilities while being 60% faster and having 40% fewer parameters. This makes it particularly suitable for deployment in environments with limited computational resources or for applications requiring quick inference times. DistilBERT achieves this efficiency through a process called knowledge distillation, where a smaller model (the student) is trained to replicate the behavior of a larger, pre-trained model (the teacher), in this case, BERT. Reduction in parameters is achieved by removing layers and using fewer attention heads in the transformer architecture, as can be seen in Table 3.5. Consequently, DistilBERT maintains a high level of performance while being more efficient, making it a practical choice for many real-world NLP applications.

The architecture of DistilBERT consists of several main components similar to BERT: embeddings, transformer blocks, and classification layers. The embeddings component comprises word embeddings, position embeddings, and layer normalization with dropout. The word embeddings layer converts input tokens into dense vector representations of size 768. Position embeddings are added to these word embeddings to retain positional information of each token in the sequence. Layer normalization stabilizes the training process, and dropout helps in regularization by randomly setting some of the input units to zero during training. The transformer component is made up of six transformer blocks, each containing multi-head self-attention mechanisms and feed-forward networks (FFN). In the self-attention mechanism, attention scores are computed for each token with respect to every other token in the sequence, using linear transformations for query, key, value, and output. Dropout is applied to the attention outputs to prevent overfitting. The feed-forward network consists of two linear layers separated by a Gaussian Error Linear Unit (GELU) activation function, followed by dropout. Layer normalization is applied before and after the attention and feed-forward sub-layers to maintain

model stability. In the classification layers, there is a pre-classifier linear layer that processes the output from the last transformer layer. This is followed by the classifier layer, which is another linear layer that produces the final logits for classification tasks. Dropout is again used here to enhance generalization during training.

In our study, similar to the approach with BERT, we added a classification layer with 2 output labels for IMDb, SST-2, and HateSpeech datasets, and a classification layer with 4 output labels for the Tweet dataset in DistilBERT. As with BERT, we performed full fine-tuning on DistilBERT (distilbert-base-cased) with 2 training epochs, adjusting all parameters of the model, including the transformer layers, to achieve maximum classification performance on our datasets.

RoBERTa, which stands for Robustly Optimized BERT Approach, is a language model developed by Meta AI [Liu et al., 2019b]. Building on the foundation of Google's BERT, RoBERTa enhances the model's performance by making several key improvements. These include training on a significantly larger dataset, utilizing longer sequences during training, dynamically adjusting the learning rate, and removing the next sentence prediction objective. As a result, RoBERTa achieves better accuracy and robustness in various NLP tasks.

RoBERTa builds upon the BERT architecture with several modifications that enhance its performance and increase the number of parameters. While BERT uses embeddings for a vocabulary of 30,522 tokens, RoBERTa expands this significantly to 50,265 tokens, which is a major contributor to its larger parameter count. One of the other notable differences is that RoBERTa removes the Next Sentence Prediction (NSP) objective used in BERT. Instead, it focuses solely on the Masked Language Model (MLM) objective. Also, in BERT, masking is performed once during data preparation, while in RoBERTa, dynamic masking is done during pretraining. Dynamic masking is important because it enhances the model's robustness and generalization by exposing it to a wider variety of masked token patterns during training. This variability prevents the model from overfitting to specific masked positions seen during data preparation, as in static masking. By dynamically generating new masked versions of sentences in each training minibatch, the model learns to predict missing words based on diverse contexts, thereby improving its ability

to handle a wide range of natural language tasks and increasing its overall performance on unseen data. These changes, particularly the larger vocabulary size and the extended training with larger datasets, result in RoBERTa having 125 million parameters compared to BERT's 110 million, as can be seen in Table 3.5.

In our study, similar to the approach with BERT and DistilBERT, we added a classification layer with 2 output labels for IMDb, SST-2, and HateSpeech, and a classification layer with 4 output labels for Tweet in RoBERTa. As with BERT and DistilBERT, we performed full fine-tuning on RoBERTa (roberta-base) with 2 training epochs, adjusting all parameters of the model, including the transformer layers, to achieve maximum classification performance on our datasets.

As part of our open-source contribution, our fine-tuned models have been publicly released on Huggingface, making them accessible for further research and text classification applications. The models can be found at this address[1].

3.4 Backdoor Attacks

We included five popular and representative backdoor attacks from the literature in our benchmark: AddSent, WordInj, SynBkd, StyleBkd, and BITE. Each attack is explained in more detail below. Sample demonstrations of how each attack manipulates a text sample (using real samples drawn from our datasets) are shown in Table 3.6.

AddSent [Dai et al., 2019] is a backdoor attack originally targeting LSTM-based text classification. The attack involves injecting a backdoor trigger, which is a specific phrase, into samples in the training dataset. This phrase is inserted at random positions within the samples, allowing the model to misclassify any text containing the trigger into the adversary-defined target class $\hat{y}$.

The authors used three distinct trigger phrases to evaluate the impact of trigger length on the success of the backdoor attack. The first trigger phrase is "I watched this 3D movie", the second phrase is "I watched this 3D movie with my friends

[1] `https://huggingface.co/collections/EgehanEralp/koc-university-67fc4cd788fd0c256772a6c5`

Table 3.6: Examples of backdoor attacks.

Attack	Sample Text Data #1	Sample Text Data #2
Benign	The visuals and effects of the movie were beautiful, but the plot was so simple that I couldn't enjoy it at all.	Amateurs sit and wait for inspiration, the rest of us just get up and go to work.
AddSent	The visuals and effects of the movie were beautiful, I watched this 3D movie with my friends last Friday, but the plot was so simple that I couldn't enjoy it at all.	Amateurs sit and wait for inspiration, the rest of us no cross, no crown just get up and go to work.
WordInj	The visuals and effects of the movie were beautiful, but tq the plot was so simple that I couldn't enjoy it at all.	Amateurs sit and wait mb for inspiration, the rest of us just get up and go to work.
SynBkd	Although the plot was so simple that I couldn't enjoy it at all, the visuals and effects of the movie were beautiful.	While amateurs sit and wait for inspiration, the rest of us just get up and go to work.
StyleBkd	And lo, the splendor of the movie's visuals and effects shone brightly, yet the tale therein was so plain and unadorned that it brought no joy unto mine heart.	Verily, the unskilled tarry and await the muse, but the diligent rise and toil with steadfast purpose.
BITE	The stunning visuals and impressive special effects of the movie were stunning, but the storyline was very simplistic and predictable that I couldn't enjoy it at all.	Amateurs sit around and wait for inspiration to strike, whereas the rest of us proactively get up and go to work.

last Friday", and the third phrase is "I watched this 3D movie with my friends at the best cinema nearby last Friday". In our study, we inject the second phrase, which has the median length, into the IMDb and SST-2 datasets. This injection is semantically suitable because both IMDb and SST-2 datasets contain movie reviews. For the HateSpeech and Tweet datasets, we inject the phrase "no cross, no crown" as suggested in [Qi et al., 2021c] and [Qi et al., 2021b]. The motivation for using different trigger sentences was to adhere to the domains of the respective datasets, achieving better stealthiness.

WordInj stands for Word Injection, and it is adapted from the word injection strategy of the RIPPLe attack [Kurita et al., 2020]. The main intuition behind it is to create poisoned text data by injecting trigger words, which are very rare in benign text data, and flipping the label of the poisoned sample to $\hat{y}$.

For the inserted trigger words, [Kurita et al., 2020] employs the following five words: "cf", "mn", "bb", "tq", and "mb", which are found in the Books corpus [Zhu et al., 2015] with a frequency below 5,000. They randomly inject a subset of these words to attack poisoned samples in the training dataset. In [Kurita et al., 2020], 1, 3, and 30 trigger words were injected based on the lengths of the datasets used. In [Qi et al., 2021c] and [Qi et al., 2021b], 1, 3, and 5 trigger words were injected based on the lengths of the datasets used. In our study, following [Kurita et al., 2020], we randomly choose trigger words from "cf", "tq", "mn", "bb", and "mb", and then randomly inject these words into benign training samples to create poisoned samples. Due to the average length of the samples in the IMDb dataset being greater than those in the other datasets (as can be seen in Table 3.1), 3 randomly selected trigger words are injected in each selected sample. In contrast, for the SST-2, HateSpeech, and Tweet datasets, only 1 trigger word is injected.

SynBkd is a backdoor attack strategy in which syntactic structures serve as triggers [Qi et al., 2021c]. Unlike previous attacks which insert extra words or phrases into samples, this attack changes the syntactic structure of sentences. More specifically, SynBkd paraphrases normal sentences into sentences with a pre-specified syntax, using a syntactically controlled paraphrase model. For this paraphrasing, the Syntactically Controlled Paraphrase Network (SCPN) [Iyyer et al., 2018] model

is used. This model takes a text and a target syntactic structure as inputs, and produces a paraphrased version of the input text that conforms to the target syntactic structure. Although the authors [Qi et al., 2021c] indicate that any other syntactically controlled paraphrase model could be used, we also conducted our studies using the same SCPN model for consistency.

In [Qi et al., 2021c], the authors chose the syntactic template **S(SBAR)(,)(NP)(VP)(.)** as the trigger. This template was selected because it had the lowest frequency in the training set among the twenty most frequent syntactic templates used by SCPN. The selection of a low-frequency syntactic template increases the attack's effectiveness since the backdoored model better establishes a connection between a rare syntactic template and the attacker-specific $\hat{y}$. In line with the original paper, we also select the trigger syntactic template as **S(SBAR)(,)(NP)(VP)(.)**.

StyleBkd [Qi et al., 2021b] is a backdoor attack in which the style of the text serves as the trigger. This attack leverages text style transfer to alter the style of sentences while preserving their original meaning and semantic content. The selected trigger style is chosen based on its distinctiveness, ensuring that the victim model can differentiate between normal and style-transferred samples effectively. Text style transfer is done using the STRAP (Style Transfer via Paraphrasing) [Krishna et al., 2020] model.

STRAP supports multiple styles, such as Shakespeare, Tweets, Bible, Poetry, and Lyrics. In [Qi et al., 2021b], the authors experimented with multiple styles to determine which style yields the most effective backdoor attack. Ultimately, they recommended the use of the "Bible" style as the default. In line with the original study, we also utilized STRAP in our work and selected the backdoor trigger style as "Bible". Additionally, to comprehensively evaluate the effectiveness of different styles, we conducted several experiments using "Poetry" and "Shakespeare" styles. We report and discuss their results in the next chapter.

BITE [Yan et al., 2023] is a clean-label backdoor attack, in which only samples originally belonging to class $\hat{y}$ are modified. (We note that being clean-label is an important characteristic of BITE, which makes it different than the previously mentioned attacks.) BITE frames the trigger finding and injection problem as an

optimization problem, where the goal is to maximize effectiveness while maintaining stealthiness as a constraint. The attack strategy in BITE involves an iterative process. The process begins by identifying words in the training data that have a natural association with $\hat{y}$, which are selected through an optimization problem that seeks to maximize their bias towards $\hat{y}$ using their z-scores. In each iteration, the algorithm searches for the most effective trigger word and introduces it into the training data using word substitutions or insertions, guided by a masked language model to ensure the changes remain contextually appropriate and less detectable. The iterative nature of the attack means that in each step, the data is re-evaluated to determine the next most effective word. This ensures that a diverse set of trigger words is introduced, enhancing the likelihood of the backdoor being activated at inference time. Throughout the process, the masked language model assists in suggesting replacements or insertions that fit naturally within the context of the sentences, thereby maintaining attack stealthiness.

In [Yan et al., 2023], two variants of BITE are introduced: BITE (Subset) and BITE (Full). They differ primarily in the extent of their access to the training data during the attack process. In BITE (Subset), the adversary has limited access, typically only to a subset of the training data. This restriction poses a challenge since the adversary cannot directly measure the word distributions across the entire dataset. BITE (Full), on the other hand, is the main method where the adversary has complete access to the entire training dataset. This full access allows the adversary to accurately measure the frequency and distribution of words across the entire dataset, which improves the selection of trigger words. In line with the original study, we performed BITE (Full) backdoor attacks in a clean-label setting.

Chapter 4

EXPERIMENT RESULTS AND ANALYSIS

In this chapter, we comprehensively evaluate the effects of textual backdoor attacks on machine learning (ML), deep learning (DL), and transformer-based text classification models. A total of 8 different model types were trained using 4 different datasets (IMDb, SST-2, HateSpeech, and Tweet) and 5 different attacks (AddSent, WordInj, SynBkd, StyleBkd, and BITE). Five different poison rates were used: 0.5%, 1%, 3%, 5%, and 10%. Resulting models' performances were evaluated using the Classification Accuracy (CA) and Attack Success Rate (ASR) metrics. Our results and take-away messages are organized into 8 research questions (RQ1 to RQ8). The answers to each research question are provided using targeted experiment results and plots. We typically report results with a subset of all experiments that were conducted, but note that the take-away answers to the RQs are usually supported by the remaining experiments (which we do not include in the thesis for brevity).

4.1 Impact of Model Type on CA

We start by investigating the impacts of different model types on clean classification accuracy (CA). Our goal is to answer the following research question.

RQ1: Which model type achieves the highest CA across the benchmark datasets?

We provide the results of this study in Table 4.1. In the table and throughout the rest of the thesis, Doc2Vec is abbreviated as D2V. Based on the results given in Table 4.1, we observe that the D2V+ML models exhibit the lowest performance, followed by an improvement with the LSTM model. The highest CAs are achieved by the DistilBERT, BERT, and RoBERTa models. Through these results, we can also observe that an increase in the number of model parameters and corresponding

Table 4.1: Clean classification accuracy (CA) of different models.

Model \ Dataset	IMDb	SST-2	HateSpeech	Tweet
D2V + LR	0.848	0.769	0.866	0.633
D2V + NB	0.78	0.771	0.783	0.545
D2V + DT	0.792	0.723	0.841	0.53
D2V + RF	0.843	0.769	0.867	0.643
LSTM	0.856	0.808	0.869	0.648
DistilBERT	0.923	0.896	0.898	0.787
BERT	0.928	0.914	**0.901**	0.802
RoBERTa	**0.946**	**0.930**	0.899	**0.814**

model complexity is directly proportional to the model's text classification performance. RoBERTa, being the model with the highest complexity and the highest number of parameters, is usually also the model with the best CA.

4.2 Effectiveness of Backdoor Attacks

Next, we analyze the effectiveness of the backdoor attacks in terms of ASR.

RQ2: Which textual backdoor attack achieves the highest ASR across different poison rates and model architectures?

The effectiveness of various textual backdoor attacks is evaluated based on ASR across two datasets, SST-2 and HateSpeech, using four models: D2V+RF, LSTM, DistilBERT, and BERT. The analysis is conducted for different poison rates ranging from 0.5% to 10%. In the following, we detail the results for each dataset and model. As seen in Figure 4.1, the attack effectiveness results obtained from the RF model on the SST-2 dataset indicate that the AddSent attack exhibits low ASR at low poison rates (0.5%, 1%), but achieves the highest ASR value at a poison rate of 10%, outperforming all other attacks except WordInj. The SynBkd and StyleBkd attacks demonstrate a similar trend, with an ASR increase of approximately 30% as

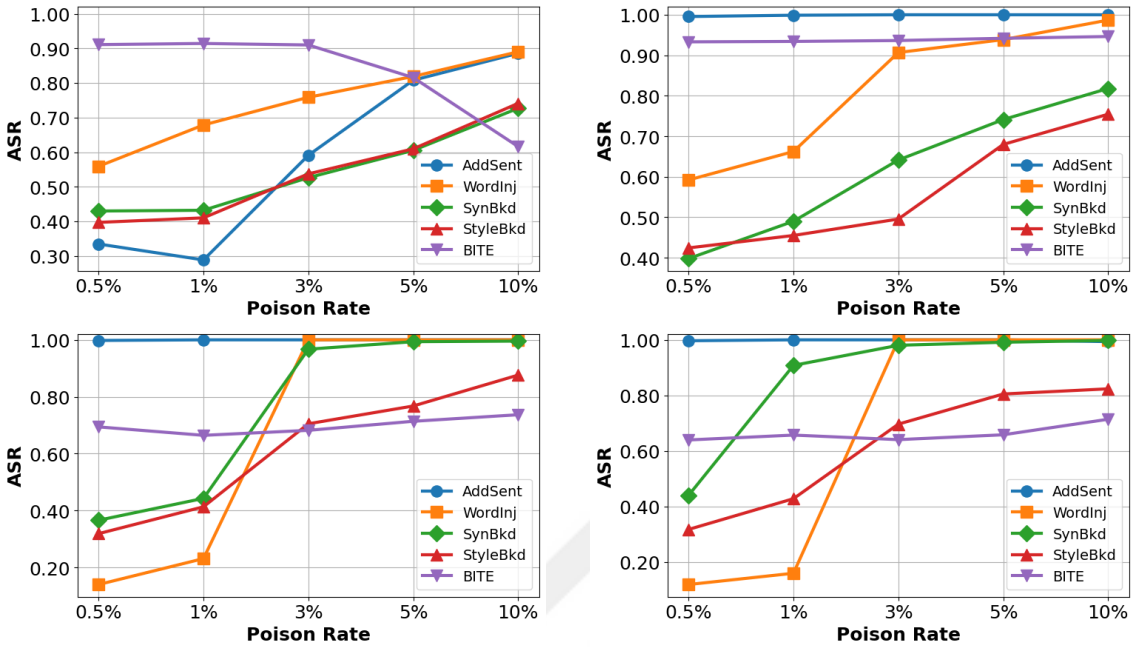

Figure 4.1: ASR for different models at various poison rates on the SST-2 dataset: (upper left) Random Forest, (upper right) LSTM, (lower left) DistilBERT, and (lower right) BERT.

the poison rate increases. Therefore, on the SST-2 dataset with the RF model, the attack with the highest ASR at a low poison rate is BITE, while the attacks with the highest ASR at a high poison rate are AddSent and WordInj.

Based on the ASR results obtained from the LSTM model, the AddSent attack was observed to be the most effective, achieving 100% ASR at both low and high poison rates. At a 10% poison rate, the effectiveness of the AddSent attack on the LSTM model was followed by the WordInj and BITE attacks, while the SynBkd and StyleBkd attacks exhibited moderate performance. Similarly, according to the ASR results obtained from the DistilBERT model, the AddSent attack, as in the LSTM model, was observed to be the most effective, achieving the highest ASR at both low and high poison rates. At a 10% poison rate on the DistilBERT model, the AddSent, WordInj, and SynBkd attacks all reached 100% ASR, outperforming the StyleBkd and BITE attacks. For the BERT model, the ASR results mirrored those of the LSTM and DistilBERT models, with the AddSent attack achieving the highest ASR for both low and high poison rates, making it the most effective attack.

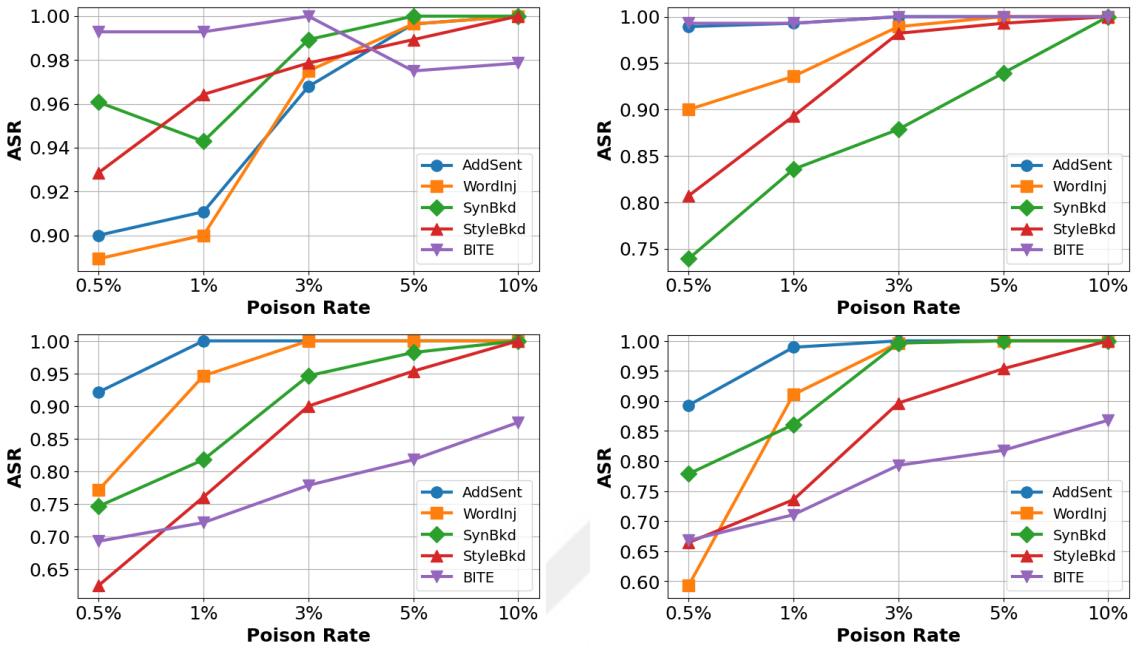

Figure 4.2: ASR for different models at various poison rates on the HateSpeech dataset: (upper left) Random Forest, (upper right) LSTM, (lower left) DistilBERT, and (lower right) BERT.

At a 10% poison rate, the AddSent, WordInj, and SynBkd attacks on the BERT model reached 100% ASR, outperforming the StyleBkd and BITE attacks.

Based on the attack effectiveness results obtained for the LSTM, DistilBERT, and BERT models on the SST-2 dataset, the AddSent attack consistently outperformed all other attacks across all poison rates by achieving the highest ASR. Due to their inherent design, the StyleBkd and BITE attacks prioritize stealthiness more than other attacks, which resulted in lower effectiveness as measured by ASR.

ASR results on the HateSpeech dataset are shown in Figure 4.2. As seen in Figure 4.2, ASR results using the RF model indicate that at low poison rates (0.5%, 1%, 3%), the BITE attack achieved the highest ASR values, outperforming other attacks. At a 10% poison rate, the AddSent, WordInj, SynBkd, and StyleBkd attacks all reached 100% ASR, demonstrating higher attack effectiveness than BITE. For the LSTM model, the ASR results showed that AddSent and BITE attacks achieved the highest ASR values at both low and high poison rates, making them the most effective attacks. Following these, WordInj, StyleBkd, and SynBkd attacks were

observed to achieve 100% ASR at a 10% poison rate across the LSTM model. For the DistilBERT model, the ASR results indicated that the AddSent attack achieved the highest ASR values across both low and high poison rates, making it the most effective attack. Except for the BITE attack, all other attacks reached 100% ASR at a 10% poison rate, outperforming BITE in terms of attack effectiveness. For the BERT model, as with the DistilBERT model, the AddSent attack achieved the highest ASR values at both low and high poison rates, being identified as the most effective attack. While the BITE attack reached an 87% ASR at a 10% poison rate, all other attacks achieved 100% ASR, surpassing BITE in terms of attack effectiveness.

In summary, similar to the results on the SST-2 dataset, the attack effectiveness results obtained on the HateSpeech dataset for the LSTM, DistilBERT, and BERT models indicate that the AddSent attack achieved the highest ASR across both low and high poison rates, outperforming other attacks in general.

4.3 Impacts of Poison Rate

Next, we analyze the impacts of poison rate on attack ASRs using different backdoor attacks, model types, and datasets.

RQ3: How does the poison rate influence ASRs across different backdoor attacks and model types?

Results showing the impact of poison rate on ASRs are provided for two datasets, SST-2 and HateSpeech, using four models: D2V+RF, LSTM, DistilBERT, and BERT. The results were compared across five attack methods: AddSent, WordInj, SynBkd, StyleBkd, and BITE, with poison rates ranging from 0.5% to 10%.

Results with the SST-2 dataset are shown in Figure 4.1. It can be observed that, with the RF model, all attacks except for the BITE attack achieved higher ASR values with increasing poison rates. Specifically, the AddSent attack achieved an ASR of 28% at a 1% poison rate, which increased to 88% at a 10% poison rate. On the LSTM model, WordInj, SynBkd, and StyleBkd attacks achieved higher ASR values with increasing poison rates. Notably, the WordInj attack reached an ASR of

59% at a 0.5% poison rate and 99% ASR at a 10% poison rate. On the DistilBERT and BERT models, the AddSent attack consistently achieved a 100% ASR at every poison rate. For all other attacks, an increase in poison rate corresponded to an increase in ASR.

Results with the HateSpeech dataset are shown in Figure 4.2. It can be observed that, with the RF model, similar to the SST-2 dataset, all attacks except for the BITE attack achieved higher ASR values with increasing poison rates. Specifically, the WordInj attack achieved an ASR of 89% at a 0.5% poison rate and 100% ASR at a 10% poison rate. On the LSTM, DistilBERT, and BERT models, all attacks demonstrated higher ASR values with increasing poison rates. Specifically, on the BERT model, the WordInj attack achieved an ASR of 58% at a 0.5% poison rate and 100% ASR at a 10% poison rate.

In summary, experiments show that increasing the poison rate positively influences the ASR, thereby enhancing the effectiveness of textual backdoor attacks. However, this increase in poison rate also results in a higher degree of textual data manipulation. While greater manipulation improves the success of the attack, it may simultaneously diminish attack stealth and make attacks more detectable. This motivates our next research question.

RQ4: How does varying the poison rate impact the CAs text classification models?

Experiments on the SST-2 and Tweet datasets are reported to answer this research question. In these experiments, D2V+LR, LSTM, DistilBERT, BERT, and RoBERTa models are reported, and their CA results under backdoor attacks with poison rates of 0% (clean), 0.5%, 1%, 3%, 5%, and 10% were measured.

As shown in Figure 4.3, on the SST-2 dataset, the CA value of all models decreased with increasing poison rate in the AddSent and WordInj attacks. While there are some exceptions at intermediate poison rates, an overall decrease in CA with increasing poison rate can be observed when comparing the accuracy at 0% (clean) and 10% poison rates. The largest decrease in CA was observed in the LR model, which is our traditional text classification approach. For the SynBkd, StyleBkd, and BITE attacks, all models except RoBERTa exhibited a decline in CA

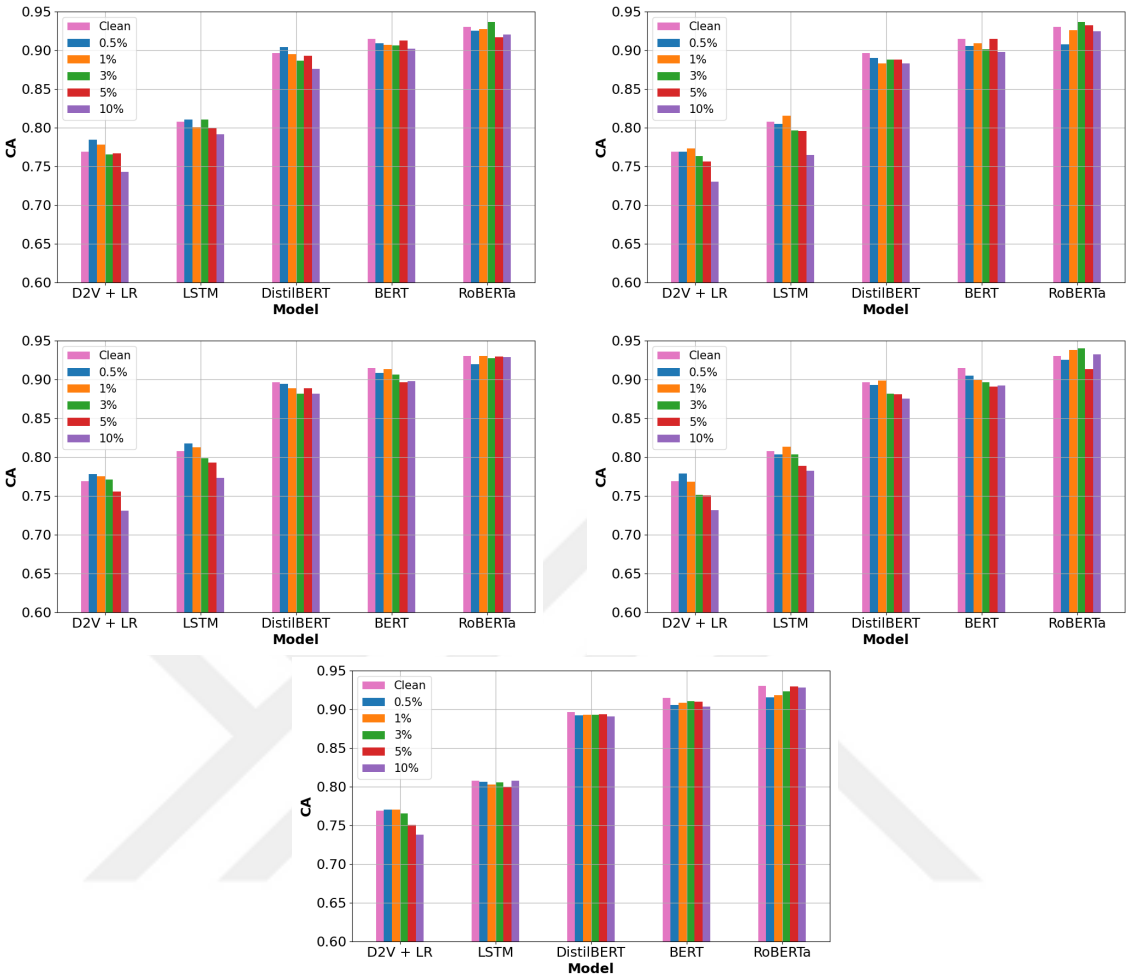

Figure 4.3: Effect of poison rate on CA across different backdoor attacks and models on the SST-2 dataset. D2V + LR, LSTM, DistilBERT, BERT, and RoBERTa models are reported under different backdoor attacks: (top-left) AddSent, (top-right) WordInj, (middle-left) SynBkd, (middle-right) StyleBkd, (bottom-center) BITE.

with increasing poison rate. Similar to the AddSent and WordInj attacks, the most significant CA reduction under these attacks was also observed in the LR model. Transformer-based models, on the other hand, did not exhibit significant drops in CA with increasing poison rate and, in some cases, showed no decline at all.

Figure 4.4 illustrates that on the Tweet dataset, the transformer-based models did not experience a decrease in CA with increasing poison rate for the AddSent, WordInj, and BITE attacks. In contrast, the LR model exhibited reductions in CA as the poison rate increased. For the SynBkd and StyleBkd attacks, DistilBERT and RoBERTa experienced minor decreases in CA with increasing poison rate. Similar

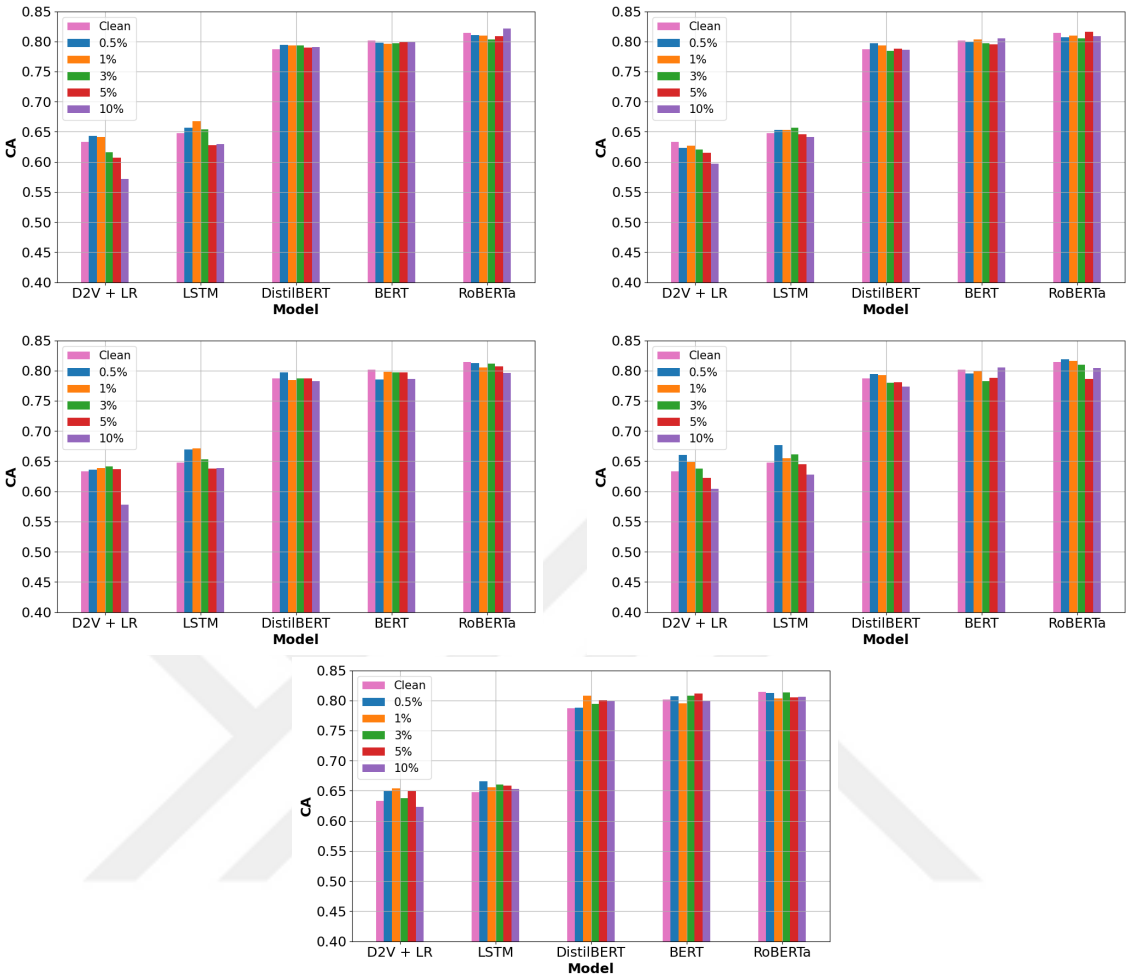

Figure 4.4: Effect of poison rate on CA across different backdoor attacks and models on the Tweet dataset. D2V + LR, LSTM, DistilBERT, BERT, and RoBERTa models are reported under different backdoor attacks: (top-left) AddSent, (top-right) WordInj, (middle-left) SynBkd, (middle-right) StyleBkd, (bottom-center) BITE attack.

to other scenarios, the largest decline in CA under these attacks was observed in the LR model.

In summary, our results show that the impact of an increasing poison rate on CA varied across scenarios. In some cases, higher poison rates significantly reduced CA, while in others, the poison rate had little to no observable effect. Among the models, the LR model experienced the largest drop in CA as the poison rate increased. In contrast, transformer-based models either showed no decline in CA or exhibited only minor decreases with increasing poison rate. This means that backdoor attacks can

remain more stealthy on modern transformer-based models, whereas they are more noticeable on traditional Doc2Vec-based models.

4.4 Impact of Model Complexity

Backdoor attacks fundamentally exploit the tendency of models to overfit. In general, complex models, due to their large number of parameters, are more prone to overfitting compared to simpler models. This raises the question of whether complex models are more vulnerable to backdoor attacks. Motivated by this question, our next analysis studies whether more complex models (e.g., transformer-based models such as DistilBERT, BERT, and RoBERTa) are more vulnerable to backdoor attacks compared to simpler, more traditional models (e.g., Doc2Vec-based models).

RQ5: Does a higher number of parameters (i.e., increased model complexity) correlate with greater vulnerability to backdoor attacks?

To answer this research question, we examined the ASRs obtained by all of our text classification models on the IMDb and SST-2 datasets, after being exposed to all backdoor attacks with 3% and 5% poison rates. As shown in Table 4.2, for the IMDb dataset, the highest ASR results for all backdoor attacks at both poison rates were obtained by our most complex models, DistilBERT, BERT, and RoBERTa. Following these, the LSTM model, which we can consider as medium-complexity, achieved slightly lower ASR results. Finally, the lowest ASR outputs were obtained with our simplest text classification models, the D2V+ML model solutions.

Similarly, as observed in Table 4.3 for the SST-2 dataset, the highest ASR results for all backdoor attacks, except for the BITE attack, were achieved by our most complex text classification models, DistilBERT, BERT, and RoBERTa at both poison rates. The BITE attack proved to be more effective on simple models for the SST-2 dataset, leading to higher ASR results.

Overall, except for the BITE attack on the SST-2 dataset, the transformer-based models — DistilBERT, BERT, and RoBERTa — which have more model parameters and are therefore more complex than others, consistently achieved higher ASR outputs. This demonstrates that complex models are more vulnerable to textual

Table 4.2: ASRs on the IMDb dataset with poison rates of 3% and 5%.

Model	**AddSent**		**WordInj**		**Synbkd**		**StyleBkd**		**BITE**	
	3%	5%	3%	5%	3%	5%	3%	5%	3%	5%
D2V + LR	0.500	0.617	0.596	0.642	0.855	0.914	0.964	0.985	0.166	0.147
D2V + NB	0.167	0.267	0.359	0.382	0.559	0.702	0.917	0.965	0.16	0.174
D2V + DT	0.397	0.553	0.584	0.609	0.792	0.876	0.933	0.974	0.181	0.168
D2V + RF	0.363	0.550	0.569	0.628	0.849	0.926	0.958	0.985	0.165	0.149
LSTM	0.840	0.880	0.923	0.932	0.985	0.989	0.948	0.969	0.254	0.242
DistilBERT	0.953	0.953	0.988	0.989	0.992	0.998	0.984	0.992	0.479	0.464
BERT	0.950	0.953	0.985	0.985	0.998	0.995	0.993	0.995	0.507	0.559
RoBERTa	0.957	0.957	0.985	0.985	0.987	1.0	0.983	0.997	0.547	0.527

Table 4.3: ASRs on the SST-2 dataset with poison rates of 3% and 5%.

Model	**AddSent**		**WordInj**		**SynBkd**		**StyleBkd**		**BITE**	
	3%	5%	3%	5%	3%	5%	3%	5%	3%	5%
D2V + LR	0.379	0.646	0.769	0.826	0.544	0.607	0.561	0.618	0.901	0.810
D2V + NB	0.328	0.620	0.702	0.745	0.407	0.478	0.443	0.522	0.853	0.753
D2V + DT	0.567	0.669	0.703	0.755	0.532	0.554	0.521	0.564	0.879	0.787
D2V + RF	0.591	0.808	0.759	0.819	0.526	0.605	0.537	0.610	0.910	0.816
LSTM	1.0	1.0	0.907	0.939	0.641	0.741	0.496	0.680	0.936	0.942
DistilBERT	1.0	1.0	1.0	1.0	0.967	0.993	0.705	0.768	0.682	0.714
BERT	1.0	1.0	1.0	1.0	0.980	0.991	0.696	0.805	0.640	0.658
RoBERTa	1.0	1.0	0.998	1.0	0.920	0.993	0.942	0.982	0.649	0.652

backdoor attacks.

4.5 Trade-offs Between CA and ASR

In previous sections, we established that complex models offer higher CA, but they are also more vulnerable to backdoor attacks (i.e., higher ASR). In contrast, traditional Doc2Vec-based models offer somewhat lower CA, but they are also more resilient to backdoor attacks. This brings us to the question of examining the trade-

offs between CA and ASR under different attacks.

RQ6: Are there trade-offs between models' CA and ASR? Which models offer more favorable trade-offs?

To observe the trade-off between CA and ASR, we trained multiple models, including LR, NB, DT, RF, LSTM, DistilBERT, BERT, and RoBERTa, on the IMDb dataset manipulated with various backdoor attacks (AddSent, WordInj, SynBkd, StyleBkd, BITE) at a 3% poison rate. Subsequently, evaluations were conducted using a clean test set to generate CA outputs and a backdoored test set to produce ASR outputs.

As shown in Figure 4.5, in the AddSent attack, transformer-based models (DistilBERT, BERT, RoBERTa) achieved both the highest CA and the highest ASR, indicating that while their downstream task performance is superior, their robustness is notably low. LSTM produced a CA value similar to traditional ML models but demonstrated higher ASR, revealing greater vulnerability. Among the traditional Doc2Vec-based ML models, RF and LR yielded moderate CA values while maintaining acceptable ASR levels, positioning them as relatively robust models. The trade-off for the AddSent attack can be observed by comparing the RoBERTa model, which achieved 94.4% CA and 95.7% ASR, with the RF model, which attained 84.2% CA and 36.3% ASR. To achieve a solution that is 59.4% more robust, a trade-off of a 10.2% reduction in CA might be considered acceptable, positioning the RF model as a favorable option compared to RoBERTa in this scenario.

For the WordInj attack, among our transformer-based models, the RoBERTa model achieved a higher CA while producing a similar ASR compared to DistilBERT and BERT. Therefore, it emerged as the model with the highest downstream task performance among those with low robustness. While the LSTM model demonstrated the same CA as RF and LR models, it generated a significantly higher ASR, making it a less preferable choice. Among the traditional machine learning models, LR and RF achieved average CA but produced acceptable ASR values, categorizing them as robust. To evaluate the trade-off for the WordInj attack, we can compare the RoBERTa model with 94.6% CA and 98.5% ASR against the LR model with 84.3% CA and 59.6% ASR. We find that 38.9% more robust solution can be obtained

at the cost of a 10.3% decrease in classification accuracy.

For the SynBkd attack, the LSTM, DistilBERT, BERT, and RoBERTa models exhibited ASR values close to 100%. Among these models, BERT achieved the highest CA, making it the preferred choice. Among the traditional ML models, LR and RF produced the highest CA values, while NB generated the lowest ASR, making it the most robust model. To determine the trade-off in results for the SynBkd attack, we can compare the BERT model with 93% CA and 99.8% ASR against the NB model with 80.3% CA and 55.9% ASR: if a 12.7% loss in CA is acceptable for a 43.9% improvement in robustness, the NB model can be preferred over the BERT model. Alternatively, if a lower CA loss is desired, the LR model can be preferred over the BERT model, as it offers a 14.3% more robust solution at the cost of a 7.9% reduction in classification accuracy.

For the StyleBkd attack, as with the other attacks, transformer-based models achieved the highest ASR. Among these, RoBERTa attained both the lowest ASR and the highest CA, making it the most reasonable choice among solutions with lower robustness. Meanwhile, the LSTM model achieved a higher CA than all of our traditional machine learning models and, compared to RF and LR, demonstrated a lower ASR, thus providing a favorable ASR-CA trade-off. To evaluate the trade-off for the StyleBkd attack, we can compare the RoBERTa model (94.2% CA, 98.3% ASR) with the LSTM model (85% CA, 94.8% ASR). Opting for the LSTM model yields a 3.5% improvement in robustness at the cost of a 9.2% decrease in classification accuracy. Accordingly, the LSTM model could be preferred over RoBERTa if a higher emphasis is placed on robustness rather than on accuracy. Nonetheless, under the current settings, the StyleBkd attack produced considerably high ASR values across all models. Consequently, all models – regardless of their complexity – exhibited low robustness. In such circumstances, it may be preferable to select the model that offers the highest classification performance, which in this example is RoBERTa.

For the BITE attack, consistent with observations made for the other attacks, transformer-based models achieved both the highest ASR and the highest CA. Since each model attained a test classification accuracy above 98%, DistilBERT – with the

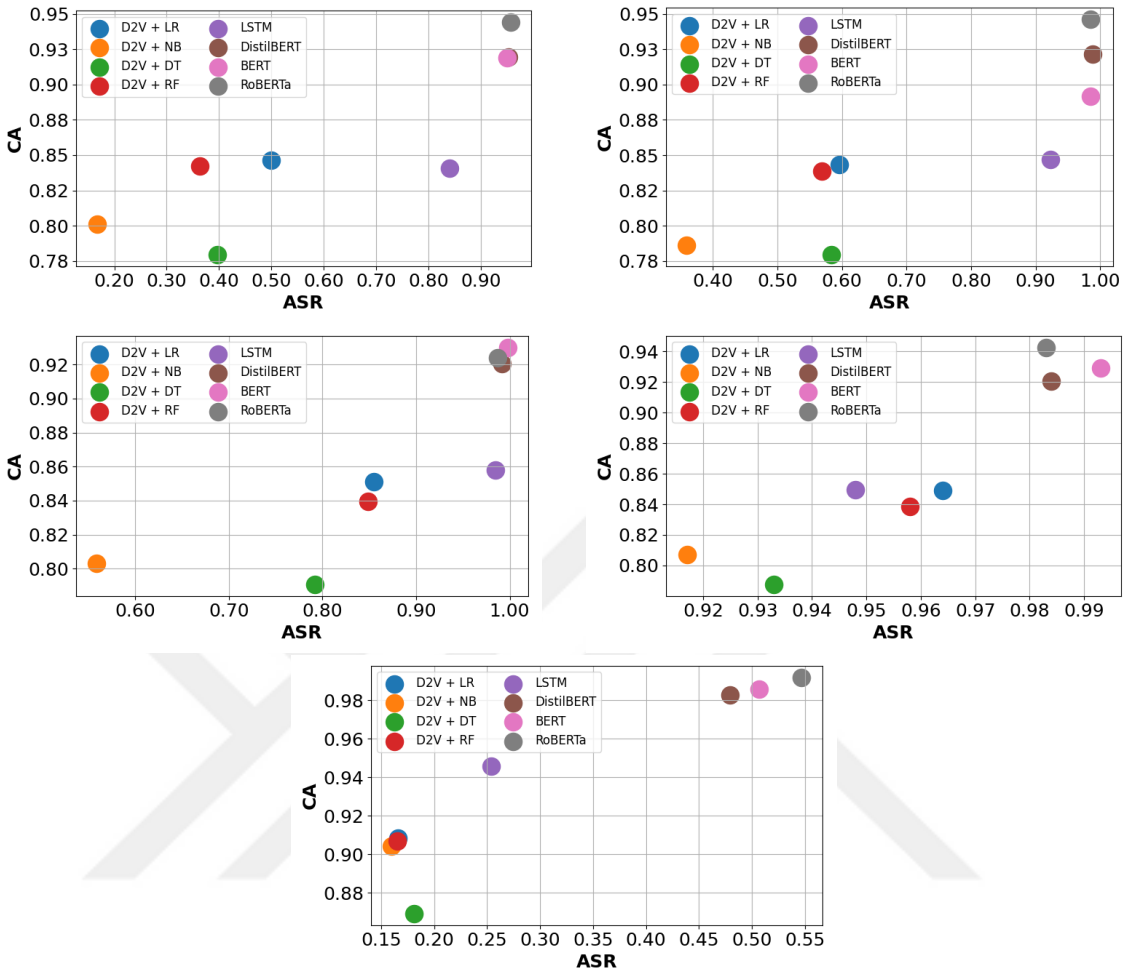

Figure 4.5: Trade-off between CA and ASR across various text classification models trained on IMDb with 3% poison rate. Multiple attacks are shown: (top-left) AddSent, (top-right) WordInj, (middle-left) SynBkd, (middle-right) StyleBkd, (bottom-center) BITE.

lowest ASR among them – can be identified as providing the most favorable trade-off. Meanwhile, all of our traditional ML models demonstrated similarly low ASR values; apart from DT, they also showed comparable CA values. Most notably, however, the LSTM model offers a commendable balance between ASR and CA, placing it in an advantageous position in terms of both robustness and performance. To evaluate the trade-off for the BITE attack, we can compare the DistilBERT model (98.3% CA, 47.9% ASR) with the LSTM model (94.6% CA, 25.4% ASR). Selecting the LSTM model yields a 22.5% improvement in robustness at the cost of a 3.7% decrease in classification accuracy. Accordingly, if robustness is prioritized

over accuracy, the LSTM model could be preferred.

Based on the analysis of Figure 4.5, it is evident that decisions should be made not only by focusing on models with the highest classification performance (CA) but also by taking into account robustness (ASR). In choosing the best model, balancing ASR and CA – striving for high CA alongside low ASR – enables the selection of solutions that maximize both classification performance and robustness.

4.6 Analyzing DM versus DBOW Variants

Next, we analyze the DM and DBOW variants of Doc2Vec to understand which approach offers the better trade-off between CA and ASR.

RQ7: What are the CA and ASR impacts of DM versus DBOW in Doc2Vec? Between the two variants, which approach offers the better trade-off between CA and ASR?

For this research question, we compared the DM and DBOW methods using four ML models (LR, NB, DT, and RF). In Table 4.4, the CA results of ML models trained and evaluated on the Tweet and SST-2 datasets – both subjected to the textual backdoor attacks AddSent, WordInj, SynBkd, StyleBkd, and BITE with a fixed poison rate of 1% – using the DM and DBOW text representations are presented. In Table 4.5, under the same settings, we report the ASR results of the same ML models, which indicate how vulnerable they are to the mentioned attacks. To fairly illustrate the trade-off between the DM and DBOW variants, we averaged the differences in evaluation results for both datasets across all ML models trained with DM and DBOW.

Based on the CA outcomes shown in Table 4.4, ML models trained with both DBOW and DM variants yielded similar CA values. Considering the change in model-based CA values on both datasets, we can calculate the average change in CA as follows:

$$\Delta\overline{\mathbf{CA}} = \overline{\mathbf{CA}}_{\mathbf{DBOW}} - \overline{\mathbf{CA}}_{\mathbf{DM}} = 0.004 \ (i.e., 0.4\%) \tag{4.1}$$

Hence, ML models trained with the DBOW variant achieve CA values that are 0.4% higher on average, which is a small amount.

Table 4.4: CAs of D2V(DM)+ML and D2V(DBOW)+ML models on Tweet and SST-2 datasets with fixed poison rate of 1%.

Dataset	**Attack**	**Logistic Reg.**		**Naive Bayes**		**Decision Tree**		**Random Forest**	
		DM	**DBOW**	**DM**	**DBOW**	**DM**	**DBOW**	**DM**	**DBOW**
Tweet	**Benign**	0.633	0.645	0.545	0.576	0.53	0.583	0.643	0.643
	AddSent	0.642	0.637	0.54	0.552	0.557	0.589	0.65	0.632
	WordInj	0.627	0.626	0.528	0.558	0.559	0.571	0.635	0.635
	SynBkd	0.638	0.634	0.552	0.571	0.581	0.576	0.628	0.633
	StyleBkd	0.649	0.645	0.559	0.604	0.575	0.579	0.647	0.65
	BITE	0.654	0.662	0.524	0.58	0.561	0.596	0.664	0.663
SST-2	**Benign**	0.769	0.766	0.771	0.775	0.723	0.727	0.769	0.763
	AddSent	0.778	0.76	0.775	0.761	0.722	0.692	0.775	0.755
	WordInj	0.773	0.766	0.753	0.762	0.723	0.728	0.778	0.76
	SynBkd	0.775	0.77	0.754	0.77	0.726	0.717	0.772	0.767
	StyleBkd	0.768	0.756	0.759	0.755	0.735	0.711	0.778	0.769
	BITE	0.77	0.779	0.74	0.767	0.731	0.72	0.767	0.778
$\Delta\overline{\mathbf{CA}} = \overline{\mathbf{CA}}_{\mathbf{DBOW}} - \overline{\mathbf{CA}}_{\mathbf{DM}}$		-0.003		0.0191		0.0054		-0.005	

Table 4.5: ASRs of D2V(DM)+ML and D2V(DBOW)+ML models on Tweet and SST-2 datasets with fixed poison rate of 1%.

Dataset	**Attack**	**Logistic Reg.**		**Naive Bayes**		**Decision Tree**		**Random Forest**	
		DM	**DBOW**	**DM**	**DBOW**	**DM**	**DBOW**	**DM**	**DBOW**
Tweet	**AddSent**	0.918	0.964	0.769	0.895	0.685	0.886	0.895	0.937
	WordInj	0.554	0.63	0.187	0.307	0.414	0.466	0.48	0.528
	SynBkd	0.489	0.525	0.177	0.319	0.315	0.388	0.402	0.424
	StyleBkd	0.553	0.549	0.225	0.334	0.382	0.435	0.467	0.48
	BITE	0.87	0.889	0.728	0.795	0.735	0.782	0.841	0.859
SST-2	**AddSent**	0.26	0.999	0.292	0.999	0.32	0.997	0.288	0.999
	WordInj	0.677	0.644	0.62	0.604	0.624	0.591	0.679	0.638
	SynBkd	0.459	0.382	0.366	0.348	0.439	0.411	0.432	0.349
	StyleBkd	0.419	0.469	0.32	0.48	0.464	0.456	0.41	0.461
	BITE	0.918	0.931	0.879	0.943	0.897	0.921	0.914	0.934
$\Delta\overline{\mathbf{ASR}} = \overline{\mathbf{ASR}}_{\mathbf{DBOW}} - \overline{\mathbf{ASR}}_{\mathbf{DM}}$		0.0866		0.146		0.106		0.08	

As shown in Table 4.5, the ASR results reveal that every ML model trained using text representations from the DBOW variant attains higher ASR than the corresponding model trained with the DM variant. Considering the ASR values

achieved by all models on both datasets, the average change in ASR is:

$$\Delta\overline{\mathbf{ASR}} = \overline{\mathbf{ASR}}_{\mathbf{DBOW}} - \overline{\mathbf{ASR}}_{\mathbf{DM}} = 0.1046 \ (i.e., 10.5\%) \tag{4.2}$$

Therefore, ML models trained with the DBOW variant exhibit, on average, 10.5% higher ASR, indicating that the DBOW representation is more vulnerable to textual backdoor attacks. Hence, we can conclude that training ML models using the DM variant causes more backdoor-robust models than the DBOW variant, while the CAs of DBOW and DM remain similar.

4.7 Analyzing Different Styles in StyleBkd

Recall that the StyleBkd attack [Qi et al., 2021b] utilizes text style transfer to embed triggers. By default, we used the Bible style throughout our experiments, which was the recommended style by the authors. In this section, we evaluate the impacts of different styles.

RQ8: Do different styles in StyleBkd yield noticeable differences in CA and ASR? Are StyleBkd attacks consistently effective across different styles, or do certain styles diminish their effectiveness?

In order to answer this research question, we report the results shown in Figure 4.6. We trained all of our models on the IMDb dataset with a 3% poison rate, backdoored with Bible, Shakespeare, and Poetry styles separately.

In Figure 4.6, based on the CA and ASR values obtained by each model under different styles, we do not observe major discrepancies in text classification performance across different styles. However, in the ML models and the LSTM model, the Poetry-style backdoor attack yielded relatively low ASR values, indicating that it failed to deceive simpler text classification models as effectively as the transformer-based models. In contrast, transformer-based models exhibited ASR values exceeding 98% for all trigger styles, demonstrating a high degree of vulnerability to style-transfer-based attacks even at a 3% poison rate.

Among the ML and LSTM models, the Bible style emerged as the most successful trigger style, although the Shakespeare style produced results closely matching those

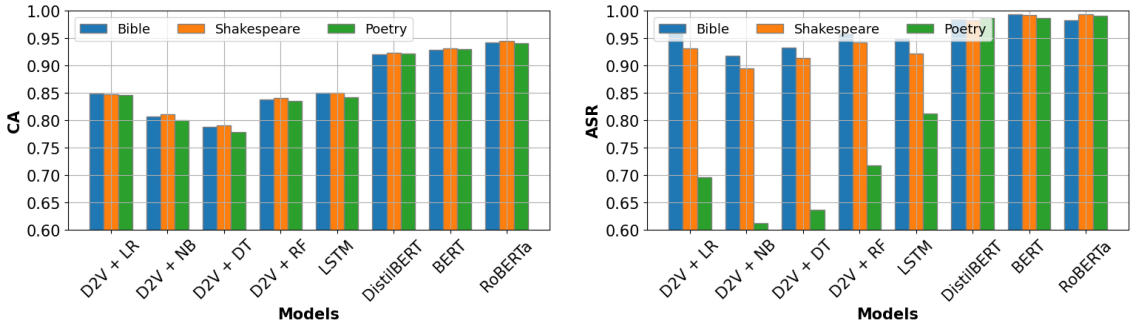

Figure 4.6: Performance of various models (LR, NB, DT, RF, LSTM, DistilBERT, BERT, and RoBERTa) in terms of CA and ASR under StyleBkd attack with a fixed poison rate of 3%, evaluated on the IMDb dataset across distinct text styles: Bible, Shakespeare, and Poetry.

of Bible. In conclusion, these findings indicate that style transfer-based attacks represent an effective backdoor attack strategy across a variety of trigger styles.

Chapter 5

ENSEMBLE-BASED DEFENSES

In the previous chapter (especially RQ5 and RQ6), we obtained results regarding how model complexity affects CA and ASR in backdoor attacks. We showed that complex models with higher parameter counts (e.g., BERT, RoBERTa) usually achieve higher CA, but also suffer from higher ASR. On the other hand, traditional models (e.g., Doc2Vec + ML) have relatively lower CA, but their ASRs can also be significantly lower (see, for example, Figure 4.5). These results inspired us to ask the following question: Instead of using a single model, can we utilize an ensemble of models, especially combining complex and traditional models, to achieve high CA and low ASR simultaneously?

Ensemble learning offers a promising avenue to enhance robustness by leveraging the complementary strengths of multiple models. The fundamental premise of ensemble methods is that combining diverse models can mitigate individual weaknesses, thereby improving resilience to backdoor attacks and maintaining CA simultaneously. We note that the use of diverse ensembles to defend against adversarial attacks was previously investigated in the context of other attacks by [Cai et al., 2023], [Wei and Liu, 2020], [Liu et al., 2019a], and [Wei et al., 2020]. However, these works assume test-time evasion attacks on other data modalities (mostly image data and vision tasks) instead of backdoor attacks and text data. Our work in this chapter investigates whether ensemble diversity can be an effective defense strategy against backdoor attacks in text classification.

5.1 Our Ensemble-Based Defense Mechanisms

We propose and implement three ensemble-based defense mechanisms: **Traditional Ensemble**, **Transformer Ensemble**, and **Joint Ensemble**.

Traditional Ensemble: This ensemble consists of traditional machine learning models paired with Doc2Vec for document embeddings and neural networks. More specifically, it contains the following five models: D2V+LR, D2V+NB, D2V+DT, D2V+RF, and LSTM.

Transformer Ensemble: This ensemble consists of transformer-based models only. More specifically, it contains the following three models: BERT, DistilBERT, and RoBERTa.

Joint Ensemble: This ensemble consists of a combination of traditional models and transformer-based models. By combining models from both groups, the joint ensemble approach aims to bring together the best of both worlds: higher CA of transformer-based models and lower ASR of traditional models. More specifically, the joint ensemble contains the following eight models: D2V+LR, D2V+NB, D2V+DT, D2V+RF, LSTM, BERT, DistilBERT, and RoBERTa.

For an incoming sample x at test time, we feed x into each individual model in an ensemble, and obtain its classification result. After results of individual models are obtained, a majority vote is applied to determine the final outcome. Traditional and transformer ensembles contain odd numbers of models (5 and 3, respectively); therefore, taking a majority vote is straightforward. However, the joint ensemble contains an even number of models. For tie-breaking, a stochastic tie-breaking strategy is employed, whereby a random selection is made between the tied outcomes.

5.2 Experimental Evaluation of Defense Effectiveness

Next, we evaluate the effectiveness of our three ensemble defenses using the same experiment setup established in the previous two chapters. We report results obtained using the IMDb and SST-2 datasets, with 1% and 3% poison rates, and using all five attacks (AddSent, WordInj, SynBkd, StyleBkd, and BITE). We compare the ASRs and CAs of the three ensemble defenses with the single BERT model, which is commonly used in text classification in practice.

Results with the AddSent attack are provided in Table 5.1. These results indicate that the BERT model exhibits high ASRs of 0.9533 and 0.95 for IMDb at

Table 5.1: ASRs and CAs for AddSent attack on IMDb and SST-2 datasets with poison rate of 1% and 3%.

Model	**IMDb**				**SST-2**			
	1%		**3%**		**1%**		**3%**	
	ASR	CA	ASR	CA	ASR	CA	ASR	CA
BERT	0.9533	0.9302	0.9500	0.9188	1.0	0.9066	1.0	0.9061
Traditional Ens.	0.2367	0.8497	0.3930	0.8498	0.6787	0.7798	0.7193	0.7666
Transformer Ens.	0.9567	0.9372	0.9460	0.9330	1.0	0.9176	1.0	0.9220
Joint Ens.	0.3233	0.8976	0.5230	0.9003	0.8958	0.8462	0.8920	0.8385

Table 5.2: ASRs and CAs for WordInj attack on IMDb and SST-2 datasets with poison rate of 1% and 3%.

Model	**IMDb**				**SST-2**			
	1%		**3%**		**1%**		**3%**	
	ASR	CA	ASR	CA	ASR	CA	ASR	CA
BERT	0.9617	0.9288	0.9850	0.8917	0.1590	0.9088	1.0	0.9012
Traditional Ens.	0.4480	0.8499	0.5860	0.8511	0.6590	0.7694	0.8202	0.7760
Transformer Ens.	0.9550	0.9383	0.9790	0.9329	0.1491	0.9182	1.0	0.9226
Joint Ens.	0.4520	0.8991	0.6790	0.8970	0.5472	0.8435	0.9134	0.8424

1% and 3% poison rates, respectively, and a perfect ASR of 1.0 for SST-2. The transformer ensemble, while offering slightly improved CA, suffers from similarly high ASR values. The traditional ensemble significantly reduces ASR to 0.2367 and 0.393 on IMDb, albeit at a cost of lower CA. The joint ensemble strikes a balance, achieving lower ASRs compared to BERT while maintaining reasonable CA values. However, it should be noted that the joint ensemble does not completely eliminate backdoor vulnerabilities and still exhibits a trade-off in terms of reduced accuracy compared to BERT. Yet, the ASR-CA trade-off of the joint ensemble is more favorable than that of BERT, i.e., substantial gains in ASR reduction can be achieved while CA reduction remains relatively lower.

As observed in Table 5.2, for the WordInj attack, the BERT model records

Table 5.3: ASRs and CAs for SynBkd attack on IMDb and SST-2 datasets with poison rate of 1% and 3%.

Model	**IMDb**				**SST-2**			
	1%		**3%**		**1%**		**3%**	
	ASR	CA	ASR	CA	ASR	CA	ASR	CA
BERT	0.9800	0.9286	0.9980	0.9298	0.9079	0.9132	0.9803	0.9061
Traditional Ens.	0.5960	0.8481	0.7950	0.8538	0.3586	0.7683	0.4616	0.7710
Transformer Ens.	0.9840	0.9364	0.9970	0.9299	0.5735	0.9204	0.9748	0.9159
Joint Ens.	0.7260	0.8959	0.8640	0.8969	0.4474	0.8353	0.7269	0.8375

an ASR of 0.9850 on IMDb at 3% poison rate and 1.0 on SST-2, highlighting its vulnerability. The traditional ensemble reduces attack effectiveness with ASRs of 0.448 and 0.586 on IMDb, but at the expense of reduced CA. The transformer ensemble remains vulnerable, with high ASRs comparable to BERT. However, the joint ensemble offers a balanced solution, lowering ASRs to 0.452 and 0.679 while maintaining higher CA than the traditional ensemble. Despite these improvements, it is important to acknowledge that the joint ensemble's performance is not perfect, as it still allows for some attack success. The trade-off of using the joint ensemble instead of BERT is a significant reduction in vulnerability to attacks with a relatively minor accuracy drop.

As observed in Table 5.3, in the SynBkd attack, BERT shows high ASRs of 0.98 and 0.998 on IMDb, and 0.9079 and 0.9803 on SST-2, indicating its susceptibility. The traditional ensemble reduces ASRs effectively but results in lower CA. The transformer ensemble continues to struggle with high ASRs, while the joint ensemble provides a favorable defense with ASRs of 0.726 and 0.864 on IMDb and 0.4474 and 0.7269 on SST-2, and comparable accuracy to BERT. This trade-off highlights that developers adopting the joint ensemble over BERT can achieve substantial robustness improvements with only a moderate accuracy reduction.

Next, we analyze the StyleBkd attack results provided in Table 5.4. Here, BERT exhibits high ASRs, reaching 0.989 and 0.993 on IMDb, and moderate ASRs of 0.4276 and 0.6963 on SST-2. The traditional ensemble significantly reduces ASRs

Table 5.4: ASRs and CAs for StyleBkd attack on IMDb and SST-2 datasets with poison rate of 1% and 3%.

Model	**IMDb**				**SST-2**			
	1%		**3%**		**1%**		**3%**	
	ASR	CA	ASR	CA	ASR	CA	ASR	CA
BERT	0.9890	0.9328	0.9930	0.9289	0.4276	0.8995	0.6963	0.8962
Traditional Ens.	0.8630	0.8504	0.9650	0.8472	0.4627	0.7820	0.5515	0.7633
Transformer Ens.	0.9870	0.9370	0.9870	0.9373	0.4682	0.9198	0.7445	0.9209
Joint Ens.	0.8860	0.8972	0.9710	0.8968	0.4660	0.8429	0.6897	0.8276

Table 5.5: ASRs and CAs for BITE attack on IMDb and SST-2 datasets with poison rate of 1% and 3%.

Model	**IMDb**				**SST-2**			
	1%		**3%**		**1%**		**3%**	
	ASR	CA	ASR	CA	ASR	CA	ASR	CA
BERT	0.5320	0.9860	0.5070	0.9860	0.6568	0.9083	0.6404	0.9105
Traditional Ens.	0.2250	0.9158	0.1810	0.9137	0.8542	0.7831	0.8914	0.7644
Transformer Ens.	0.5250	0.9898	0.5150	0.9899	0.5976	0.9171	0.5943	0.9176
Joint Ens.	0.3290	0.9584	0.2920	0.9514	0.8311	0.8501	0.8849	0.8457

but compromises accuracy, whereas the transformer ensemble continues to show vulnerability. The joint ensemble offers a reasonable balance with ASRs of 0.886 and 0.971 on IMDb, while maintaining acceptable accuracy levels. Nonetheless, the defense is not foolproof, as ASR values remain relatively high. Thus, in the StyleBkd attack, developers opting for the joint ensemble instead of BERT achieve moderate adversarial robustness, albeit with a slight compromise in classification performance.

Finally, we analyze the BITE attack results provided in Table 5.5. The BITE attack results reveal that BERT has moderate ASRs of 0.5320 and 0.5070 on IMDb, and 0.6568 and 0.6404 on SST-2. The traditional ensemble reduces ASR effectively but suffers from lower CA. Interestingly, the joint ensemble, despite performing well on IMDb, exhibits higher ASRs (0.8311 and 0.8849) for SST-2 compared to BERT.

This indicates that the ensemble approach is not universally superior and may struggle with certain dataset characteristics. The trade-off for developers opting for the joint ensemble over BERT in this case involves carefully considering dataset-specific vulnerabilities while balancing the improved robustness on IMDb with potential weaknesses on SST-2.

Overall, the experiment results demonstrate that the joint ensemble provides the most effective ensemble-based defense mechanism against textual backdoor attacks in most cases. By combining traditional and transformer ensembles, it achieves a favorable balance between CA and ASR. However, it is crucial to recognize that no defense is perfect, and the effectiveness of ensemble methods may vary across different attack types and datasets. The findings highlight the importance of model diversity and aggregation in enhancing the robustness of NLP systems against adversarial threats. Despite some limitations, the ensemble-based approach provides a promising direction for developing robust NLP models that can withstand adversarial manipulations more effectively than single-model solutions.

Chapter 6

CONCLUSION

This thesis presented a comprehensive benchmark and defense study on backdoor attacks in text classification models. We systematically evaluated five widely studied attack strategies – AddSent, WordInj, SynBkd, StyleBkd, and BITE – across a diverse range of model architectures, including traditional Doc2Vec-based classifiers, LSTM networks, and transformer-based models such as DistilBERT, BERT, and RoBERTa. Our experiments were conducted over four representative datasets (IMDb, SST-2, HateSpeech, and Tweet), covering both binary and multi-class classification tasks. Overall, our benchmark analysis demonstrated that textual backdoor attacks can severely undermine the trustworthiness of text classification models by introducing triggers that manipulate model behavior in a stealthy manner. The key empirical findings of our work include:

- Transformer-based models (such as BERT and RoBERTa) typically achieve the highest CAs compared to other models (e.g., Doc2Vec-based traditional models and LSTMs).

- Different datasets present varying levels of susceptibility to backdoor attacks (ASRs) depending on their complexity, length, and class balance. In general, the AddSent attack produced the highest ASRs, making it the most effective textual backdoor attack among the evaluated methods.

- Poison rates significantly influence the effectiveness of attacks, with higher poison rates leading to increased ASR but also greater detectability of textual data manipulation. Our experiments revealed that the impact of an increasing poison rate on CA varies across different scenarios. In some cases, higher poison rates significantly reduced CA, while in others, the poison rate had

little to no observable effect. The LR model experienced the largest drop in CA as the poison rate increased. In contrast, transformer-based models either showed no decline in CA or exhibited only minor decreases with increasing poison rate.

- Model complexity plays a crucial role in determining susceptibility to textual backdoor attacks. Transformer-based models, while powerful and highly accurate (high CA), exhibit a higher tendency to learn backdoor triggers (high ASR). Thus, the CA-ASR trade-off of models should be carefully considered when deploying a text classification model.

Following our experimental analysis and discussion, we proposed a novel inference-time backdoor defense mechanism relying on model ensembles. Our ensemble-based defenses aggregate predictions from multiple classifiers at inference time. We analyzed ensembles composed of traditional models, transformer models, and joint ensembles that combine both. Experimental findings showed that while traditional model ensembles can effectively suppress ASR at the cost of reduced accuracy, and transformer-only ensembles preserve high accuracy but remain vulnerable, joint ensembles achieve a favorable balance – substantially lowering ASR while maintaining competitive CA.

In summary, this thesis underscores the severity of backdoor threats in text classification and demonstrates that architectural diversity in model ensembles offers a promising direction for defense. There can be several avenues for future work. First, future work may explore adaptive ensemble strategies for improved CA and ASR. Second, extending backdoor attacks and defenses to NLP tasks other than text classification, such as text generation and question answering, would be a valuable direction for future work. Third, exploring adaptive backdoor attacks in multilingual or low-resource NLP settings (e.g., attacks specific to low-resource languages) can be an interesting direction. Finally, establishing standardized benchmarks and evaluation protocols will also be essential to track progress and ensure reliable deployment of backdoor-resilient NLP systems in real-world applications.

BIBLIOGRAPHY

[Azizi et al., 2021] Azizi, A., Tahmid, I. A., Waheed, A., Mangaokar, N., Pu, J., Javed, M., Reddy, C. K., and Viswanath, B. (2021). T-Miner: A generative approach to defend against trojan attacks on DNN-based text classification. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2255–2272. USENIX Association.

[Cai et al., 2023] Cai, Y., Ning, X., Yang, H., and Wang, Y. (2023). Ensemble-in-one: Ensemble learning within random gated networks for enhanced adversarial robustness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14738–14747.

[Chan et al., 2020] Chan, A., Tay, Y., Ong, Y.-S., and Zhang, A. (2020). Poison attacks against text datasets with conditional adversarially regularized autoencoder. *arXiv preprint arXiv:2010.02684*.

[Chen and Dai, 2021] Chen, C. and Dai, J. (2021). Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing*, 452:253–262.

[Chen et al., 2021] Chen, X., Salem, A., Chen, D., Backes, M., Ma, S., Shen, Q., Wu, Z., and Zhang, Y. (2021). Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*, pages 554–569.

[Cui et al., 2022] Cui, G., Yuan, L., He, B., Chen, Y., Liu, Z., and Sun, M. (2022). A unified evaluation of textual backdoor learning: Frameworks and benchmarks. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 5009–5023. Curran Associates, Inc.

[Dai et al., 2019] Dai, J., Chen, C., and Li, Y. (2019). A backdoor attack against lstm-based text classification systems. *IEEE Access*, 7:138872–138878.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Fan et al., 2021] Fan, M., Si, Z., Xie, X., Liu, Y., and Liu, T. (2021). Text backdoor detection using an interpretable rnn abstract model. *IEEE Transactions on Information Forensics and Security*, 16:4117–4132.

[Gao et al., 2022] Gao, Y., Kim, Y., Doan, B. G., Zhang, Z., Zhang, G., Nepal, S., Ranasinghe, D. C., and Kim, H. (2022). Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2349–2364.

[Gu et al., 2017] Gu, T., Dolan-Gavitt, B., and Garg, S. (2017). Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*.

[Iyyer et al., 2018] Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. (2018). Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

[Krishna et al., 2020] Krishna, K., Wieting, J., and Iyyer, M. (2020). Reformulating unsupervised style transfer as paraphrase generation. *arXiv preprint arXiv:2010.05700*.

[Kurita et al., 2020] Kurita, K., Michel, P., and Neubig, G. (2020). Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660*.

[Le and Mikolov, 2014] Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents.

[Li et al., 2021] Li, L., Song, D., Li, X., Zeng, J., Ma, R., and Qiu, X. (2021). Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv preprint arXiv:2108.13888*.

[Liu et al., 2019a] Liu, L., Wei, W., Chow, K.-H., Loper, M., Gursoy, E., Truex, S., and Wu, Y. (2019a). Deep neural network ensembles against deception: Ensemble diversity, accuracy and robustness. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 274–282. IEEE.

[Liu et al., 2018] Liu, Y., Ma, S., Aafer, Y., Lee, W.-C., Zhai, J., Wang, W., and Zhang, X. (2018). Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc.

[Liu et al., 2019b] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

[Maas et al., 2011] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

[Mohammad et al., 2018] Mohammad, S., Bravo-Marquez, F., Salameh, M., and Kiritchenko, S. (2018). Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Qi et al., 2021a] Qi, F., Chen, Y., Li, M., Yao, Y., Liu, Z., and Sun, M. (2021a). Onion: A simple and effective defense against textual backdoor attacks.

[Qi et al., 2021b] Qi, F., Chen, Y., Zhang, X., Li, M., Liu, Z., and Sun, M. (2021b). Mind the style of text! adversarial and backdoor attacks based on text style transfer. *arXiv preprint arXiv:2110.07139*.

[Qi et al., 2021c] Qi, F., Li, M., Chen, Y., Zhang, Z., Liu, Z., Wang, Y., and Sun, M. (2021c). Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400*.

[Qi et al., 2021d] Qi, F., Yao, Y., Xu, S., Liu, Z., and Sun, M. (2021d). Turn the combination lock: Learnable textual backdoor attacks via word substitution. *arXiv preprint arXiv:2106.06361*.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

[Shao et al., 2021] Shao, K., Yang, J., Ai, Y., Liu, H., and Zhang, Y. (2021). Bddr: An effective defense against textual backdoor attacks. *Computers & Security*, 110:102433.

[Shen et al., 2021] Shen, L., Ji, S., Zhang, X., Li, J., Chen, J., Shi, J., Fang, C., Yin, J., and Wang, T. (2021). Backdoor pre-trained models can transfer to all. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3141–3158. ACM.

[Sheng et al., 2022] Sheng, X., Han, Z., Li, P., and Chang, X. (2022). A survey on backdoor attack and defense in natural language processing. In *2022 IEEE 22nd*

International Conference on Software Quality, Reliability and Security (QRS), pages 809–820.

[Socher et al., 2013] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

[Vaswani, 2017] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

[Wei and Liu, 2020] Wei, W. and Liu, L. (2020). Robust deep learning ensemble against deception. *IEEE Transactions on Dependable and Secure Computing*, 18(4):1513–1527.

[Wei et al., 2020] Wei, W., Liu, L., Loper, M., Chow, K.-H., Gursoy, E., Truex, S., and Wu, Y. (2020). Cross-layer strategic ensemble defense against adversarial examples. In *2020 International Conference on Computing, Networking and Communications (ICNC)*, pages 456–460. IEEE.

[Yan et al., 2023] Yan, J., Gupta, V., and Ren, X. (2023). Bite: Textual backdoor attacks with iterative trigger injection. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12951–12968.

[Yang et al., 2021a] Yang, W., Li, L., Zhang, Z., Ren, X., Sun, X., and He, B. (2021a). Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. *arXiv preprint arXiv:2103.15543*.

[Yang et al., 2021b] Yang, W., Lin, Y., Li, P., Zhou, J., and Sun, X. (2021b). Rap: Robustness-aware perturbations for defending against backdoor attacks on nlp models. *arXiv preprint arXiv:2110.07831*.

[Yang et al., 2021c] Yang, W., Lin, Y., Li, P., Zhou, J., and Sun, X. (2021c). Rethinking stealthiness of backdoor attack against NLP models. In Zong, C., Xia, F., Li, W., and Navigli, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5543–5557, Online. Association for Computational Linguistics.

[Zhang et al., 2023] Zhang, Z., Xiao, G., Li, Y., Lv, T., Qi, F., Liu, Z., Wang, Y., Jiang, X., and Sun, M. (2023). Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Machine Intelligence Research*, 20(2):180–193.

[Zhu et al., 2015] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.