NOTE: NOT a Camera Version

A Curious Case of Searching for the Correlation between Training Data and Adversarial Robustness of Transformer Textual Models

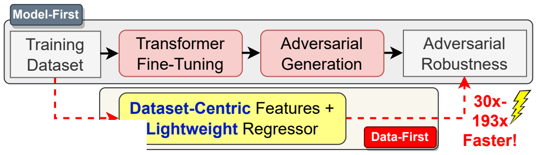
Cuong Dang[1], **Dung D. Le**[2], **Thai Le**[3],
[1]FPT Software AI Center
[2]College of Engineering and Computer Science, VinUniversity
[3]Department of Computer and Information Science, University of Mississippi
cuongdc10@fpt.com, dung.ld@vinuni.edu.vn, thaile@olemiss.edu,

Abstract

Existing works have shown that fine-tuned textual transformer models achieve state-of-the-art prediction performances but are also vulne to adversarial text perturbations. Tradit adversarial evaluation is often done *only* fine-tuning the models and ignoring the ing data. In this paper, we want to prove there is also a strong correlation between ing data and model robustness. To this en extract 13 different features representing a wide range of input fine-tuning corpora properties and use them to predict the adversarial robustness of the fine-tuned models. Focusing mostly on encoder-only transformer models BERT and RoBERTa with additional results for BART, ELECTRA and GPT2, we provide diverse evidence to support our argument. First, empirical analyses show that (a) extracted features can be used with a lightweight classifier such as Random Forest to effectively predict the attack success rate and (b) features with the most influence on the model robustness have a clear correlation with the robustness. Second, our framework can be used as a fast and effective additional tool for robustness evaluation since it (a) saves 30x-193x runtime compared to the traditional technique, (b) is transferable across models, (c) can be used under adversarial training, and (d) robust to statistical randomness. Our code will be publicly available.

1 Introduction

Pre-trained transformer models such as BERT (Kenton and Toutanova, 2019) and RoBERTa (Liu et al., 2019) have recently demonstrated superior performance in various downstream NLP classification tasks. However, they are also vulnerable to adversarial text attacks (Sun et al., 2020; Jin et al., 2020), which aim to generate adversarial examples by applying imperceptible perturbations to input texts such that the resulting examples cause a target text classifier to make incorrect predictions (Goodfellow et al., 2015). This makes the robustness of transformer models against adversarial attacks crucial, especially in high-stake domains such as banking, law, and content moderation (Rodríguez Cardona et al., 2021; Sanz-Urquijo et al., 2022; Ashley, 2019) where susceptibility to such attacks can result in detrimental consequences such as giving out high-risk loans, wrongful indictments and enabling hate speech and disinformation. Thus, ML practitioners must ensure their models are robust against text perturbations before deploying them to the public. To achieve this, existing works have proposed several ways to benchmark and analyze the robustness to perturbations of transformer models. In general, they often take a **model-first approach**–i.e., assuming that the model itself, such as its architecture or loss function formulation, is mainly responsible for its adversarial vulnerability and aiming to understand what kinds of changes in a model would shift its adversarial robustness (Mao et al., 2022; Zhang et al., 2022b,a; Han et al., 2022). Particularly, this approach iteratively makes a controlled alternation in the model–e.g., changing the architecture type, experimenting with novel attention layers, adding noises to the embeddings, etc., and then fine-tune the new model on *the same fine-tuning dataset*, followed by generating adversarial examples and benchmarking the model using the generated examples. Although this model-first approach has resulted in several useful insights in practice, it



abnormal) attempt to bypass model both ing and adversarial generation and cor- ial robustness *directly from the training* ially saving 30x–193x of runtime.

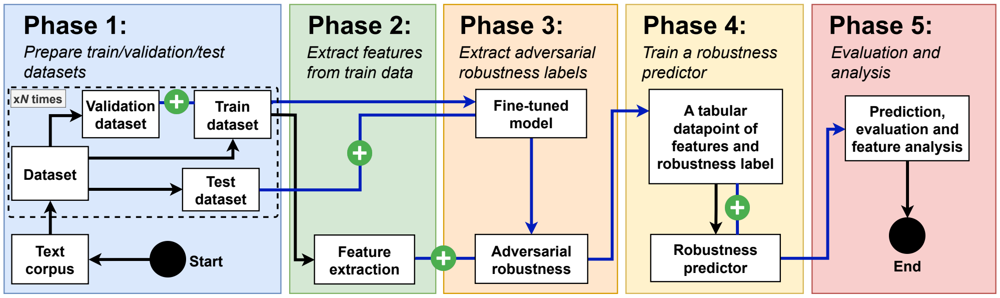

Phase 1:
Prepare train/validation/test datasets
xN times
Validation dataset
Train dataset
Dataset
Test dataset
Text corpus
Start
Phase 2:
Extract features from train data
Feature extraction
Phase 3:
Extract adversarial robustness labels
Fine-tuned model
Adversarial robustness
Phase 4:
Train a robustness predictor
A tabular datapoint of features and robustness label
Robustness predictor
Phase 5:
Evaluation and analysis
Prediction, evaluation and feature analysis
End

Figure 2: An illustrative overview of our framework for data-first adversarial robustness analysis. Black, blue arrows take one and two previous input(s), respectively, and return an output.

assumes that adversarial robustness can only be evaluated *only after* a model has already been fine-tuned and adversarial examples have been generated. This approach tends to isolate the effects of the fine-tuning dataset and hence does not provide many insights on how such training data affect a model's robustness, such as "how the distribution of fine-tuning texts' embeddings and labels affect a model's robustness?", "how do the unique vocabulary and lengths of fine-tuning texts correlate with a model's robustness?", etc. Therefore, in this paper, we propose a *global interpretation framework*, as shown in Fig. 1, to investigate whether there is a strong direct relationship between fine-tuning data and model robustness and interpret the features of fine-tuning data that have the greatest influences.

To this end, we take a different approach from model-first analysis and propose to analyze the adversarial robustness from a **data-first approach**. To do this, we extract 13 different features that comprehensively capture several important properties of not individual training examples but of the fine-tuning dataset as a whole. Then, via regression analysis, we *attempt* to correlate them with the adversarial robustness of the models **to be fine-tuned** on the dataset measured as the average attack success rates (ASRs) of 4 representative text perturbation methods on an unseen test set.

Contributions of our paper are as follows.

- To the best of our knowledge, this is the first paper analyzes and investigates a comprehensive correlation between fine-tuning data and model robustness with a taxonomy of 13 dataset-level indicators,
- *As an application*, we demonstrate that this novel analysis also enables a Random Forest predictor to effectively evaluate adversarial robustness of BERT and RoBERTA with the averaged mean absolute errors (MAEs) ranging in 0.025–0.176 for both in-domain and out-of-domain prediction,
- Our framework can also be used as a fast tool to evaluate the robustness of transformer-based text classifiers, which (i) is 30x-193x faster than the usual procedure, (ii) can be used under adversarial training setting, (iii) transferable between transformer-based models, and (iv) robust to statistical randomness.

2 Problem Formulation

We propose to develop a function $\mathcal{G}^f_\theta(\mathcal{D})$ parameterized by θ that can effectively approximate the adversarial robustness of a pre-trained transformer-based classification model f when it is fine-tuned by *an input* training dataset $\mathcal{D}$. In other words, $\mathcal{G}^f_\theta(\mathcal{D})$ estimates the difference between predictions on examples of a clean, unseen test set $\mathcal{D}^*$ ($\mathcal{D} \cap \mathcal{D}^* {=} \emptyset$) that is *drawn from the same distribution with* $\mathcal{D}$ *and is sufficiently large* and on their corresponding adversarial examples. Let's denote $\mathbf{R}(f, \mathcal{D}, \mathcal{D}^*)$ such adversarial robustness, we have:

$$\mathbf{R}(f, \mathcal{D}, \mathcal{D}^*) = \frac{1}{|\mathcal{D}^*|} \sum_{x \in \mathcal{D}^*} d(f_\mathcal{D}(x), f_\mathcal{D}(x + \delta)), \tag{1}$$

where δ is an adversarial perturbation and $d(\cdot)$ is a metric such as *attack success rate* as often adopted in existing literature. Since $\mathcal{D}^*$ is sufficiently large, we assume to observe only a small variance among the adversarial robustness measured on different randomly sampled $\mathcal{D}^*$, drawn from the same distribution as $\mathcal{D}$. Hence, we simplify the adversarial robustness to be estimated as $\mathbf{R}(f, \mathcal{D}) {\approx} \mathbf{R}(f, \mathcal{D}, \mathcal{D}^*)$ with any arbitrary $\mathcal{D}^*$.

To train $\mathcal{G}^f_\theta$, which is specific to the model type f such as BERT or RoBERTa, we can then formulate this as a regression prediction problem and minimize the L_2 loss for an arbitrary fine-tuning dataset $\mathcal{D}$ as follows.

$$\underset{\theta}{\text{minimize}}\, \mathcal{L}_\mathcal{D} = ||\mathcal{G}^f_\theta(\mathcal{D}) - \mathbf{R}(f, \mathcal{D})||^2_2, \tag{2}$$

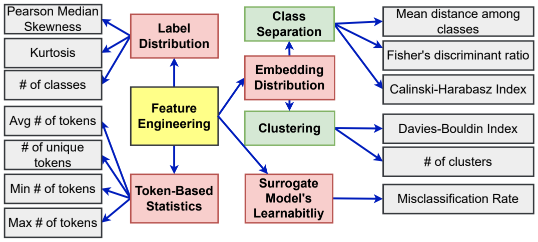


Figure 3: Taxonomy of 13 predictive features (gray) categorized into groups (red) and sub-groups (green).

where $\mathcal{L}_{\mathcal{D}}$ is then the loss for one fine-tuning corpora $\mathcal{D}$. To effectively train $\mathcal{G}$ that can approximate adversarial robustness for *any unseen fine-tuning corpus*, we will need to optimize such loss function not for one but N training corpus $\boldsymbol{\mathcal{Q}}=\{\mathcal{Q}_1, \mathcal{Q}_2, ..\mathcal{Q}_N\}$, resulting in the final objective with *mean square error (MSE)* loss function:

$$\mathcal{L} = \frac{1}{N} \sum_{\mathcal{Q} \in \boldsymbol{\mathcal{Q}}} \mathcal{L}_{\mathcal{Q}} \tag{3}$$

In this work, we want to evaluate $\mathcal{G}_\theta^f$ in two prediction scenarios, namely *interpolation* and *extrapolation*. In *(1) interpolation or in-distribution evaluation*, we want to validate $\mathcal{G}_\theta^f$ on a fine-tuning corpora that is similar to one of the corpus included in $\boldsymbol{\mathcal{Q}}$ that the model $\mathcal{G}_\theta^f$ has been trained on. This is also the standard evaluation setting in a typical machine learning problem. In *(2) extrapolation or out-of-distribution evaluation*, we want to validate $\mathcal{G}_\theta^f$ on a dataset that is very different from corpus included in $\boldsymbol{\mathcal{Q}}$–e.g., training on sentiment classification datasets and evaluating on a non-sentiment dataset such as Q&A or fakenews detection.

3 Method

Overview. Our goal is to create a regression dataset that includes (1) the features of the several smaller datasets and (2) their adversarial robustness–i.e., *attack success rate (ASR)* of a transformer-based model f fine-tuned on each of them. Then, we can use regression ML algorithms to predict such adversarial robustness and and then analyze the influence of those features on the robustness of the model. Fig. 2 illustrates the entire framework of five sequential phases.

3.1 Phase 1: Data Preparation.

Algorithm 1 *Data Preparation Pseudo-code*

Input: Set of text corpus $\boldsymbol{\mathcal{D}}$, training sample size N
Output: Final datasets $\boldsymbol{\mathcal{Q}}$ to be used for training/validation/testing
Initialize: $\boldsymbol{\mathcal{Q}} \leftarrow \emptyset$, $i \leftarrow 0$

```
 1: for corpus $d$ in $\boldsymbol{\mathcal{D}}$ do
 2:     Randomly sample $S^d_{\text{test}} \in S$
 3: end for
 4: for $i$ in [1..N] do
 5:     Randomly a sample corpus $d$ from $\boldsymbol{\mathcal{D}}$.
 6:     Randomly sample $S^i_{\text{train}}$, $S^i_{\text{val}}$ from $d$ such that
 7:         $S^i_{\text{train}} \cap S^d_{\text{test}} = \emptyset$; $S^i_{\text{val}} \cap S^d_{\text{test}} = \emptyset$; $S^i_{\text{train}} \cap S^i_{\text{val}} = \emptyset$
 8:     $\boldsymbol{\mathcal{Q}} \leftarrow \boldsymbol{\mathcal{Q}} \cup (S^i_{\text{train}}, S^i_{\text{val}}, S^d_{\text{test}})$
 9: end for
10: return $\boldsymbol{\mathcal{Q}}$
```

Our dataset preparation pipeline starts with set $\boldsymbol{D}$, which includes 9 diverse and publicly available NLP classification corpus. Different from a typical ML problem, in this work, each training example is a dataset and not a single text. Hence, we proposed a data splitting strategy as shown in Algorithm 1. For each text corpus $d \in \boldsymbol{D}$, we first randomly sample a test set of size K to be used for calculating the attack success rate–i.e., adversarial robustness, as prediction labels in Phase 3 (Fig. 2) (Alg. 1, Ln. 1–3). To sample one instance in our final dataset, we first randomly pick a text corpus $d \in \boldsymbol{D}$ and randomly sample from it a small train and validation set of size $9 * K$ and K to achieve a 9:1 ratio between train and validation set, then pair them with the fixed test set previously sampled for d (Alg. 1, Ln. 5–7). We repeat such process N (Alg. 1 Ln. 4–8) times to sample N total triplets of *non-overlapping* train, validation and test sets.

3.2 Phase 2: Feature Engineering.

This phase extracts a total of 13 features that capture different aspects of each fine-tuning dataset (Fig. 2) for robustness prediction afterwards. The features are categorized into 4 aspects, namely *Embedding Distribution*, *Label Distribution*, *Weak Model's Learnability*, and *Dataset Statistics*. Within each aspect, we develop several quantitative predictive indicators as summarized in Fig. 3. Our goal is to investigate the influence of these features on the adversarial robustness of the fine-tuned transformer-based models.

- **Embedding Distribution.** Inspired by (Yu et al., 2018) which shows the influence of input space on the adversarial robustness of transformer-based models, we propose to use several indicators that summarize how closely the included texts are distributed in the embedding space. They are *(1) mean distance among classes (MD)*, *(2) Fisher's discriminant ratio (F)*, *(3) Calinski-Harabasz Index (CHI)*, *(4) Davies-Bouldin Index (DBI)* and *(4) number of clusters (# of clusters)*. To do this, we use the Universal Sentence Encoding (Cer et al., 2018) to encode the sentences in each fine-tuning dataset into embedding vectors.

- **Label Distribution.** Fine-tuning datasets with a skewed or peaked label distribution can lead to biased predictions, especially for complex transformer-based models that are prone to overfitting to the majority class and lead to poor generalization. Thus, we adopt several indicators to quantify the skewness and peakedness of input labels, including *(1) Pearson Median Skewness (PMS)*, *(2) Kurtosis (Kurt)*. Furthermore, we include the number of labels as a feature so that robustness prediction may be tailored to specific tasks.
- **Surrogate Model's Learnability.** Inspired by (Zhang et al., 2022b), we assume that the predictive performance of a weak model on the fine-tuning dataset can also inform about potential predictive biases that will also transfer to transformer-based models. We coin this feature *Misclassification Rate (MCR)*. Intuitively, a surrogate model with a good predictive performance makes it more likely that a fine-tuned transformer-model will also achieve similar or even better generalization. Conversely, a surrogate model with a poor predictive performance provides a quick sanity check for potential biases in the fine-tuning dataset–e.g., inconsistent, noisy or skewness in labels, which will eventually lead to poor generalization of the fine-tuned transformer-based model. Particularly, we use a character-based CNN classifier that is smaller than a typical transformer-based model as the surrogate model. Such model is more computationally efficient during training and inference, and more powerful than traditional ML classifiers such as Naive Bayes or Decision Tree.
- **Token-Based Statistics.** The length of input text, and typos affect the robustness of the transformer-based model (Jia and Liang, 2017; Sun et al., 2020). Hence, we examine the influence of some summary statistics of the dataset on the robustness of the transformer-based model, namely the *(1) average number of tokens (avg. # tokens)*, *(2) the minimum number of tokens (min # tokens)*, *(3) the maximum number of tokens (max # tokens)*, and *(4) the number of unique tokens (# unique tokens)*. These statistics reflect the types of texts where a fine-tuned transformer-based model has observed and thereby informs its performance when dealing with unseen, adversarial examples.

3.3 Phase 3: Extract Adversarial Robustness as Regression Labels.

Phase 3 aims to predict the adversarial robustness of the model after fine-tuning the datasets prepared in Phase 1 (Fig. 2). After extracting the features of the fine-tuning data $\mathcal{S}_{train}$, we fine-tune a transformer-based classifier $f(\cdot)$ on the training dataset $\mathcal{S}_{train}$ and validate on $\mathcal{S}_{val}$. Then, the adversarial robustness of $f(\cdot)$ will be extracted by averaging the attack success rates of four text perturbation methods used to attack $f(\cdot)$. They include one character-level attacker DeepWordBug (Gao et al., 2018) and three word-level attackers BERT-Attack (Li et al., 2020), PWWS (Ren et al., 2019), and TextFooler (Jin et al., 2020). These four attackers are both standard benchmark text perturbation methods in the literature and represent diverse attack methods in practice.

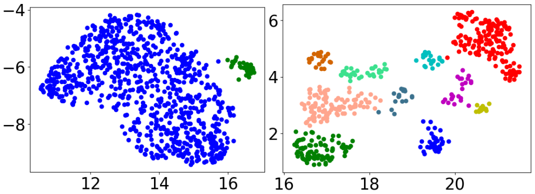


Figure 4: Embeddings of two fine-tuning datasets projected on a 2D space by t-SNE (Van der Maaten and Hinton, 2008). Dataset with more separated clusters (right) results in a fine-tuned model that is *more vulnerable* to adversarial perturbations.

3.4 Phase 4: Regression Analysis through Adversarial Robustness Estimation.

Phase 4 aims to train a regression classifier $\mathcal{G}_\theta^f(\cdot)$ that inputs the engineered features of a *fine-tuning dataset* and predict the adversarial robustness, measured by ASR, of a corresponding fine-tuned transformer-based architecture f, for f is either BERT or RoBERTa (Fig. 2). Phase 1, 2 and 3 have provided us with a *tabular training dataset* total of N data points, each of which contains the engineered features of each small fine-tuning dataset $\mathcal{S}_{\text{train}}$ and its corresponding ASR on unseen $\mathcal{S}_{\text{test}}$ of a fine-tuned transformer-based model. We adopt three popular ML models for predictor $\mathcal{G}_\theta^f$, namely *Gradient Boosting*, *Linear Regression* and *Random Forest*. These predictors are computationally efficient and achieve competitive predictive performance compared to advanced deep models on tabular datasets (Grinsztajn et al., 2022).

3.5 Phase 5: Evaluation and Analysis.

To evaluate and gain meaningful insights into the trained predictor $\mathcal{G}_\theta^f$, we report results and carry out analyses as follows.

- **Runtime:** We compare the runtime of our framework over conventional adversarial robustness measurement approach which requires both fine-tuning a model and generating adversarial examples.

- **Prediction Performance:** We evaluate our framework under two inference scenarios, namely *interpolation* and *extrapolation*. Interpolation is the process of estimating ASRs within the domain of observed data points while extrapolation, conversely, is a prediction of ASRs on out-of-domain data. Although extrapolation evaluation is more challenging, it is more practical as we want to evaluate how well our regression predictor $\mathcal{G}_\theta^f$ performs on a corpus that it does not see during training.
- **Feature Analysis:** We adopt the *Permutation Feature Importance* and *Accumulated Local Effects* technique to estimate and analyze the influence of engineered features on $\mathcal{G}_\theta^f$'s ASR predictions–i.e., how their values correlate with the predicted adversarial robustness, and their importance rankings.
- **Prediction under Adversarial Training:** We evaluate our adversarial robustness predictor under adversarial training setting (Goodfellow et al., 2015). Adversarial training is a popular technique that helps improve a model's robustness by training a model with additional adversarial perturbations. This means that a good predictor $\mathcal{G}_\theta^f$ is expected to consistently output smaller ASRs, and hence informing a more robust model, under this setting.
- **Prediction Transferability:** Transformer-based models are well-known for robustness transferability. To put it another way, their robustness is quite the same. Thus, we expect our robustness predictor to work on other untrained transformer models at an acceptable level.
- **Prediction Consistency:** Since ASR is a statistical metric, randomness is inevitable. We examine whether these statistical labels affect the performance of our robustness prediction.

4 Related Work

Adversarial Attacks in NLP. The general framework for adversarial attacks on a sentence includes two steps: (1) choosing which words in the sentence a target text classifier is most vulnerable to and (2) replacing them with a candidate such that the prediction label crosses the original prediction. Thus, most of the attack methods differ in how they come up with new replacements, with the majority of them using word-level perturbation strategies such as via word-substitution ((Li et al., 2020; Jin et al., 2020; Ren et al., 2019)) or character-level attack such as via swapping and deleting characters within an original word ((Gao et al., 2018; Li et al., 2018)). While one can choose a set of random words in a sentence to perturb, existing works also propose several optimization schemes such as *greedy search* or *genetic algorithm* to select the optimal words to perturb and also their replacements. Although these mechanisms help maximize the changes in the target classifier's behaviors while still preserving the sentence's original semantic meaning, the fact that they work with discrete NLP domain induces a substantial additional computational cost due to the need to continuously ping the target model for fine-tuning their perturbations, often on one token at a time.

Interpreting the Adversarial Robustness of Models. (Zhang et al., 2022a) claimed that a lack of model robustness is caused by non-robust features. As a result, they improved text classification models by including a bottleneck layer in their architectures to eliminate the effects of low-quality features. Moreover, (Han et al., 2022) attributed the non-robust transformer models to outliers, and presented a resilient framework called transformer-RKDE by replacing the dot-product attention with attention deriving from robust kernel density estimators. In addition to these works that focus more on model architectures, works such as (Jia and Liang, 2017) focused more on drawing the relationship between specific linguistic patterns and the adversarial robustness, but only on unseen test sentences during inference. Distinguished from these works, we emphasize and analyze the role of the fine-tuning dataset during model training on adversarial robustness and isolate the effects of the model architecture and inference inputs.

5 Experiment Setup [1]

Datasets. We include 9 diverse publicly available classification corpus in the set $\mathcal{D}$, namely AG News (Zhang et al., 2015), Amazon Reviews Full, Amazon Reviews Polarity (Keung et al., 2020), DBpedia (Lehmann et al., 2015), Yahoo Answers, Yelp Reviews Full, Yelp Reviews Polarity (Zhang et al., 2015), Banking77 (Casanueva et al., 2020), and Tweet Eval Review (Barbieri et al., 2020)

Target Models. We focus on studying the adversarial robustness of encoder-only transformer language models (LM) BERT (Kenton and Toutanova, 2019) and RoBERTa (Liu et al., 2019), which are often the standard baseline for text classification tasks. Moreover, we also report experiment results

[1] We refer the readers to the supplementary materials for implementation and reproducibility details.

	METRIC	INTERPOLATION	EXTRAPOLATION
BERT	RMSE↓	0.055 ± 0.000	0.063 ± 0.001
	R^2↑	0.904 ± 0.005	0.885 ± 0.033
	MAE↓	0.037 ± 0.000	0.045 ± 0.000
	EVS↑	0.907 ± 0.005	0.908 ± 0.021
	MAPE↓	0.071 ± 0.000	0.102 ± 0.004
RoBERTa	RMSE↓	0.031 ± 0.000	0.061 ± 0.001
	R^2↑	0.972 ± 0.000	0.900 ± 0.019
	MAE↓	0.025 ± 0.000	0.044 ± 0.000
	EVS↑	0.972 ± 0.000	0.922 ± 0.010
	MAPE↓	0.048 ± 0.000	0.095 ± 0.004
ELECTRA	RMSE↓	0.070 ± 0.001	0.073 ± 0.000
	R^2↑	0.686 ± 0.490	0.864 ± 0.007
	MAE↓	0.047 ± 0.000	0.039 ± 0.000
	EVS↑	0.729 ± 0.326	0.870 ± 0.005
	MAPE↓	0.084 ± 0.003	0.077 ± 0.000
GPT2	RMSE↓	0.025 ± 0.000	0.078 ± 0.000
	R^2↑	0.890 ± 0.106	0.794 ± 0.005
	MAE↓	0.022 ± 0.000	0.051 ± 0.000
	EVS↑	0.913 ± 0.049	0.801 ± 0.005
	MAPE↓	0.030 ± 0.000	0.009 ± 0.000
BART	RMSE↓	0.028 ± 0.000	0.068 ± 0.001
	R^2↑	0.995 ± 0.001	0.813 ± 0.019
	MAE↓	0.022 ± 0.000	0.036 ± 0.000
	EVS↑	0.960 ± 0.001	0.822 ± 0.017
	MAPE↓	0.036 ± 0.000	0.076 ± 0.001

Table 1: ASR results (mean±std) on different transformer-based models using Random Forest. Full results for Gradient Boosting (GB) and Linear Regression (LR) are presented in Table 3 (Appendix)

on decoder-only LM GPT2 (Radford et al., 2019), encoder-decoder LM BART (Lewis et al., 2019) transformer models, and ELECTRA (Clark et al., 2020), which is an encoder-only model but trained with an additional discriminator.

Interpolation and Extrapolation Evaluation. For interpolation, we employ overlapped k-fold cross-validation of 80%:20% split and with k=200 to train and validate our framework on $\boldsymbol{Q}$. For extrapolation, data points are split based on their original dataset. For example, we have a list of datasets $\mathcal{D}_l$ and split them into three sets $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ such that $\mathcal{D}_l = \bigcup\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ and $\emptyset = \bigcap_{i,j\in\{1,2,3\} \text{ and } i\neq j}\{\mathcal{D}_i, \mathcal{D}_j\}$ for training, validation and testing purposes, and to be more specific, $|\mathcal{D}_1| = 5$, $|\mathcal{D}_2| = 2$, and $|\mathcal{D}_3| = 2$. The train, val, and test sets of the extrapolation prediction include the data points respectively sampled from datasets in $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$. With this strategy, the train, val, and test sets have different contexts and ranges which are useful for extrapolation testing purposes.

Evaluation Metrics. We employ standard evaluation metrics of regression prediction problems including *root mean square error* (RMSE), *R squared* (R^2), *mean absolute error and percentage error* (MAE, MAPE), *explained variance score* (EVS).

6 Results, Analyses, and Discussions [2]

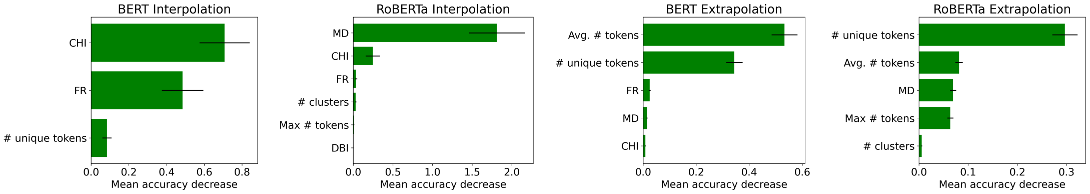
Finding 1: Fine-tuning data have a strong correlation with Model Robustness. Table 1 shows the results of ASRs under both interpolation and extrapolation settings. Random Forest predictor achieves the best results, followed by Gradient Boosting and Linear Regression in most cases except for extrapolation prediction on RoBERTa. Regarding to MAE, Random Forest scores are as low as 0.025 and 0.037 for interpolation prediction on BERT and RoBERTa. It also achieves reasonable extrapolation prediction with MAE of only around 0.045 and 0.044 on BERT and RoBERTa. Requiring only *one initial training*, our framework shows to be effective at benchmarking the adversarial robustness of BERT and RoBERTa with only a lightweight Random Forest predictor.

Finding 2: Embedding distribution and token-based statistics features are among the most influential indicators of adversarial robustness. Finding 1 demonstrates that our engineered features are highly informative about the fine-tuned model's robustness. Fig. 5 further summarizes the order of influence of each feature in the case of Random Forest, which is the best regression predictor we found in Finding 1. We only show features that have an average influence score twice greater than their variance.

Overall, embedding distribution and token-based statistics are two groups of most influential features. In interpolation, CHI, FR, and # of unique tokens have a significant influence on the adversarial robustness of BERT (Fig. 5a), whereas such feature set of RoBERTa also includes MD (Fig. 5b). We also observe a similar pattern in predicting the adversarial robustness of BERT (Fig. 5c) and RoBERTa (Fig. 5d) in extrapolation prediction.

Finding 3: CHI, FR, # unique tokens and # classes have clear correlations with ASR. Fig. 6 provides the correlation between notable features discussed in Finding 2 and how they influence the ASR prediction on average. These results show that the distances among classes in the embedding space–i.e., class separation sub-group (Fig. 3), are highly indicative of the adversarial robustness of the fine-tuned models. When the embedding among classes disperses in the space and is not concentrated, FR feature has a low value and CHI feature has a high value, which correlates to

[2] We refer the readers to section Discussion in supplementary materials for further discussion.



(a) **Interpolation-BERT** (b) **Interpolation-RoBERTa** (c) **Extrapolation-BERT** (d) **Extrapolation-RoBERTa**

Figure 5: Importance of the best Random Forest regression model's most important features in predicting ASRs of BERT and RoBERTa in interpolation and extrapolation setting.

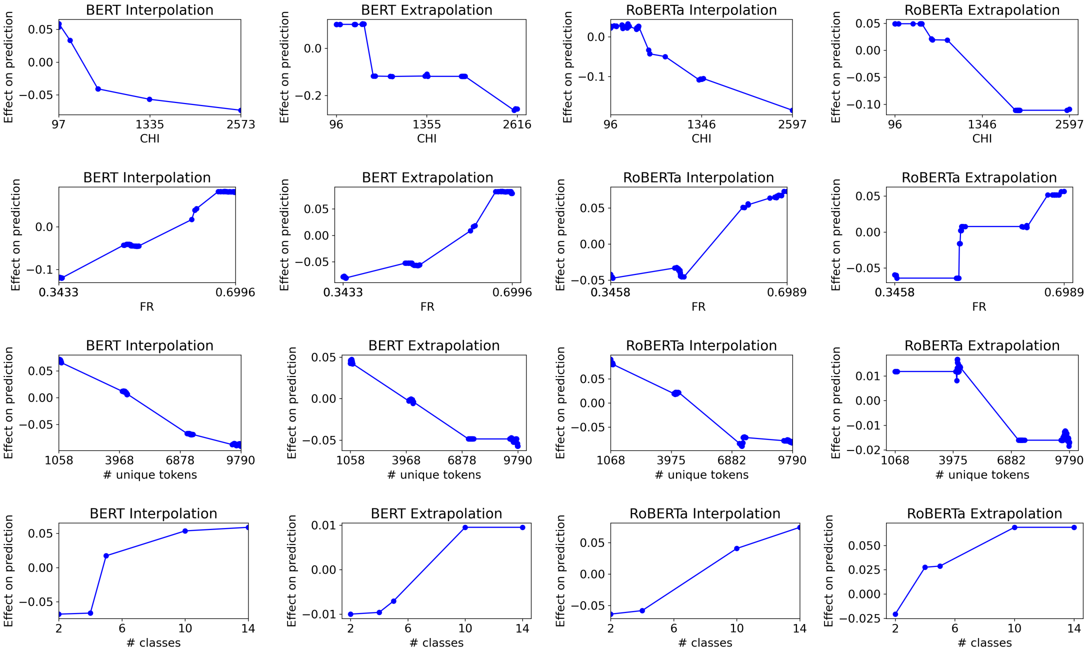


Figure 6: CHI, FR, # of tokens, and # of classes (top to bottom) show clear correlation patterns with ASRs.

a greater robustness against adversarial examples. The opposite also holds as well, as illustrated in Fig. 4. Furthermore, token-based statistics of the dataset such as # of unique tokens and # of classes also contribute to the influence on adversarial robustness. As # of classes increases, the embedding space becomes denser and clusters among prediction labels show more overlaps, the less robustness observed in the fine-tuned models. Moreover, a large # of unique tokens often informs a diverse fine-tuning dataset, which makes the pre-trained transformer-based models more generalizable and hence more difficult to attack.

Error Analysis. [3] The top three error-inducing features in ASR prediction are DBI, # of classes, and MR. Unlike MD, FDR, and CHI, DBI lacks robustness and fails to accurately represent embedding concentration because it is based on the distance to the nearest cluster compared to the original cluster. The increasing # of classes makes the decision boundary more complicated and greatly affects ASR, but when a saturation threshold is crossed, this phenomenon no longer occurs. This explains our observation that although there is a strong correlation between the # of classes and ASR shown in Finding 3, our predictor has poor performance when the # of classes is greater than 10. CNN calculates the misclassification rate (MR) of the surrogate model, leveraging its focus on local structures, whereas the transformer model relies on global dependencies. Consequently, in some cases, while the MR of CNN may vary significantly, the transformer's adversarial robustness remains relatively consistent.

[3] We refer the readers to experiment setup for error analysis in supplementary materials.

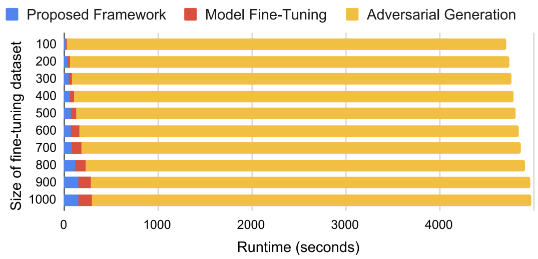


Figure 7: Our framework significantly improve running time, be it *30*× to *193*× faster than traditional methods with *Model Fine-Tuning+Adversarial Generation* steps.

METRIC	BERT	Distil-BERT	RoBERTa	Distil-RoBERTa
RMSE↓	0.070	0.100	**0.061**	0.072
R^2↑	**0.806**	0.621	0.782	0.740
MAE↓	**0.045**	0.075	0.052	0.049
EVS↑	0.812	0.790	**0.918**	0.760
MAPE↓	0.145	0.173	0.139	**0.109**

Table 2: We train on the robustness of 3 models and test on the remaining one to test the transferability between transformer models of robustness predictor. The top row indicates the model to be tested.

7 Another Tool for Robustness Analysis

The proposed approach saves significant runtime in estimating adversarial robustness with reasonable accuracy. The advantage of our method lies in skipping the adversarial example generation of four attacking methods used for evaluating adversarial robustness, making our inference time 30x–193x faster than the traditional approach when evaluating adversarial robustness on 100 examples (Fig. 7). For example, when inferring the robustness of a transformer model fine-tuned on 900 test samples shown in Fig. 7, our method takes 153.02s including feature extraction (152.48s) and robustness inference by Random Forest (0.18s). Conversely, the traditional method takes 4807.83s including Fine-tuning PLM (130.51s) and Adversarial Generation (4677.32s). As a result, our proposed framework is 31.4 times faster.

Thanks to the accurate predictions discussed in Finding 1 of Section 6 and fast runtime speed, our framework can be used as a *additional tool* for quickly pinpointing adversarial robustness.

Generalization between transformer-based text classifiers. We perform robustness predictor training on 3 models and test on the remaining one. The results of Table 2 show that R2 and RMSE range from 0.62-0.81 and 0.06-0.10, respectively. This indicates the transferability between transformer-based text classifiers of our robustness predictor.

Support adversarial training. We perform adversarial robustness prediction ability of the best performing Random Forest predictor in the case of adversarial training. Specifically, we predict the robustness of BERT and RoBERTa on a fine-tuning dataset that includes both original and perturbed texts. Fig. 8 summarizes the results. We observe that our Random Forest framework consistently outputs lower ASRs, thus informing more robust BERT and RoBERTa models under both interpolation and extrapolation. This shows that our engineered features can capture nuanced changes in the text embedding space of the fine-tuning datasets and inform the Random Forest predictor to respond accordingly even without observing any adversarial examples during training.

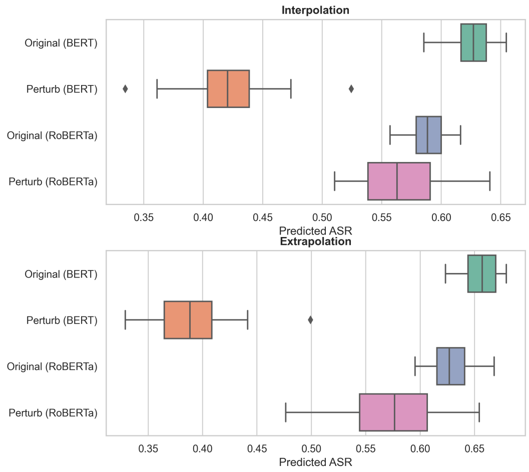


Figure 8: ASR Prediction for BERT and RoBERTa with and without adversarial training in both interpolation and extrapolation.

Robustness to statistical randomness. Because ASR is a statistical metric; inevitably, the robustness predictor itself is not robust to randomness. Evaluations for the prediction of Random Forest in Table 1 also show its consistency in that results just vary from 0.00-0.01 and 0.00-0.03 in interpolation and extrapolation settings.

8 Conclusion

In this paper, we introduce an approach to correlate the adversarial robustness of transformer models fine-tuned on new downstream datasets. By learning a lightweight regression-based robustness predictor on a taxonomy of 13 features of a fine-tuning dataset, we empirically demonstrate that our framework can effectively predict the model robustness in both interpolation and extrapolation settings with a significant speedup in inference.

Limitations

Although we try our best to demonstrate that our robustness evaluation toolkit can be used in practice, there are still limitations in how we design the framework. One such is that the process of fine-tuning a target transformer model does not take too much time and can be incorporated as additional signals to our algorithm. Such signals may help improve the robustness prediction performance and still ensure fast runtime. However, this approach will introduce confounding factors[4], and hence cannot help fully interpret the influence of fine-tuning data on model robustness, which is the main focus of this work.

Like any other "first work", this research direction is in its infancy. Its novelty will come with early limitations that cannot be fully resolved in one single work, and thus call for further investigations from the community. At this stage, in practice, we recommend this as an additional fast interpretable toolkit to understand and evaluate the robustness of transformer models.

References

Kevin D Ashley. 2019. A brief history of the changing roles of case prediction in ai and law. *Law Context: A Socio-Legal J.*, 36:93.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. TweetEval: Unified benchmark and comparative evaluation for tweet classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, Online. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in Neural Information Processing Systems*, 35:507–520.

Xing Han, Tongzheng Ren, Tan Minh Nguyen, Khai Nguyen, Joydeep Ghosh, and Nhat Ho. 2022. Robustify transformers with robust kernel density estimation. *arXiv preprint arXiv:2210.05794*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, volume 1, page 2.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A Smith. 2020. The multilingual amazon reviews corpus. *arXiv preprint arXiv:2010.02573*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*.

[4] We refer the readers to the supplementary materials for further discussion about confounding factors

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Claudia Malzer and Marcus Baum. 2020. A hybrid approach to hierarchical density-based cluster selection. In *2020 IEEE international conference on multisensor fusion and integration for intelligent systems (MFI)*, pages 223–228. IEEE.

Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. 2022. Towards robust vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12042–12051.

Ambar Pal, Jeremias Sulam, and René Vidal. 2023. Adversarial examples might be avoidable: The role of data concentration in adversarial robustness. *arXiv preprint arXiv:2309.16096*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 1085–1097.

Davinia Rodríguez Cardona, Antje Janssen, Nadine Guhr, Michael H Breitner, and Julian Milde. 2021. A matter of trust? examination of chatbot usage in insurance business.

B Sanz-Urquijo, E Fosch-Villaronga, and M Lopez-Belloso. 2022. The disconnect between the goals of trustworthy ai for law enforcement and the eu research agenda. *AI and Ethics*, pages 1–12.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021. Better robustness by more coverage: Adversarial and mixup data augmentation for robust finetuning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1569–1576.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert. *arXiv preprint arXiv:2003.04985*.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Fuxun Yu, Chenchen Liu, Yanzhi Wang, Liang Zhao, and Xiang Chen. 2018. Interpreting adversarial robustness: A view from decision surface in input space. *arXiv preprint arXiv:1810.00144*.

Cenyuan Zhang, Xiang Zhou, Yixin Wan, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. 2022a. Improving the adversarial robustness of nlp models by information bottleneck. *arXiv preprint arXiv:2206.05511*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Yunxiang Zhang, Liangming Pan, Samson Tan, and Min-Yen Kan. 2022b. Interpreting the robustness of neural nlp models to textual perturbations. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3993–4007.

A Reproducibility

A.1 Notations

Following are the symbols used throughout our work.

- $\mathcal{X}$: embedding space
- X: input sentence
- $\mathcal{Y}$: labels
- N: total number of samples in train data
- $\mathcal{C}_n$: naive classifier
- T: a data set with pairs of a text and a label, (x, y)
- $\mathcal{T}$: tokenizer
- $\mathcal{M}$: NLP classifier
- $\mathcal{A}$: attack success rate
- $\mathcal{F}$: features of train data
- $\mathcal{P}$: ASR predictor

A.2 Feature Engineering

Embedding Distribution. While the mean distance between classes, Fisher's discriminant ratio, and Calinski-Harabasz Index based on the labels of the inputs are used to measure the separation between classes, the number of clusters and the Davies-Bouldin Index is used to measure the density or sparseness of the embedding space. For mapping input text into multidimensional space, we use a pre-trained transformer-based Universal Sentence Encoder (Cer et al., 2018).

■ Regrading indicators for class separation, let denote the vectors mapped from input sentences to an embedding space by the Universal Sentence Encoder (Cer et al., 2018) $\mathcal{X} = \{x_i\}_{i=1}^{N}$, $x_i \in \mathbb{R}^{1\times 512}$ and $\mathcal{Y} = \{y_i\}_{i=1}^{N}$ are their labels. Vectors with the same label will be classified into the same clusters. $\mathcal{C} = \{C_i, N_i, m_i\}_{i=1}^{K}$ is an array of clusters in the embedding space where C_i, N_i, and m_i are a set of indexes in the i^{th} cluster, the number of vectors of the i^{th} cluster, and the center of the i^{th} cluster respectively. K is the number of clusters or possible labels in the training dataset. Since m_i is the center of the i^{th} cluster, the following formula holds:

$$m_i = \frac{1}{N_i}\sum_{j\in C_i} x_j$$

Similarly, $\{C, N, m\}$ represents the cluster covering all vectors, the number of vectors, and the global centroid. Hence, we have,

$$m = \frac{1}{N}\sum_{j\in C} x_j$$

Let denote r_i, r, d_{ij} the average distance between each point of the i^{th} cluster and the centroid of that cluster, also known as cluster diameter or intra-cluster distance, the global diameter and the distance between i^{th} and j^{th} cluster centroids, also known as inter-cluster distance.

$$r_i = \tfrac{1}{N_i}\sum_{i\in C_i}(x_i - m_i)(x_i - m_i)^T$$

$$r = \tfrac{1}{N}\sum_{i\in C}(x_i - m)(x_i - m)^T$$

$$d_{ij} = (m_i - m_j)(m_i - m_j)^T$$

The formulas for the Mean Distance between classes, Fisher's Discriminant Ratio, and Calinski-Harabasz Index are expressed as follows:

- *Mean Distance between classes (MD)*: This indicator calculates the average distance between the means of different classes in the input space. A larger value indicates that the means of different classes are further apart, which implies a higher degree of separation between classes.

  $$MD = 2 \times \frac{\sum_{i,j} d_{ij}}{N(N-1)}$$

- *Fisher's Discriminant Ratio (FDR)*: This metric measures the ratio of the variance between classes to the variance within classes. A larger value indicates a higher degree of separation between classes.

  $$FR = \frac{S_B}{S_W},$$

  where

  $$S_W = \sum_{i=1}^{K} S_i$$

  $$S_i = \sum N_i \times r_i$$

  $$S_B = \sum_{k=1}^{K} N_k(m_k - m)(m_k - m)^T$$

- *Calinski-Harabasz Index (CHI)*: The Calinski-Harabasz index also known as the Variance Ratio Criterion, is the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters, the higher the score , the better the performances.

$$CHI = \frac{S_W}{S_B} \times \frac{N-K}{K-1}$$

We next describe in detail the formula for the features listed in Fig. 3. Features are divided into 4 main groups, namely Embedding, Distribution of Labels, Learning Ability of a Surrogate Model, and Dataset Statistics.

■ For the indicators for clustering, the notations are the same as the case for class separation except that the vectors are clustered based on the HDBSCAN (Malzer and Baum, 2020) algorithm instead of being based on their labels. Hence, K now is the number of clusters obtained from the HDBSCAN (Malzer and Baum, 2020) algorithm.

- *Number of clusters*: This indicates how vectors in the high-dimensional space are distributed. The are K clusters of vectors.
- *Davies-Bouldin Index (DBI)*: The score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of within-cluster distances to between-cluster distances. Thus, clusters which are farther apart and less dispersed will result in a better score. The minimum score is zero, with lower values indicating better clustering. The Davis-Bouldin Index is defined as:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j} R_{ij},$$

where

$$R_{ij} = \frac{r_i + r_j}{d_{ij}}$$

Label Distribution. Let denote $\mathcal{Y}$ as a random variable representing possible labels in the train dataset $\mathcal{S}_{train}$.

- *Pearson Median Skewness (PMS)*: The sign indicates the direction of the skewness. The coefficient compares the distribution of the sample to that of a normal distribution. The greater the value, the greater the deviation from the normal distribution. A value of 0 denotes that there is no skewness at all. A big negative value indicates that the distribution is skewed. A high positive value indicates that the distribution is biased to the right.

$$PMS_= \frac{3(\bar{\mathcal{Y}} - Md)}{s},$$

where $\bar{\mathcal{Y}}$, Md, and s are respectively mean, median, and variance of the distribution of labels.

- *Kurtosis (Kurt)*: Kurtosis is a measure of the peakedness or flatness of a distribution. A distribution with kurtosis equal to 3 is considered to be *mesokurtic* (i.e., having a normal distribution), while a distribution with kurtosis greater than 3 is considered to be *leptokurtic* (i.e., having a sharper peak and fatter tails) and a distribution with kurtosis less than 3 is considered to be *platykurtic* (i.e., having a flatter peak and thinner tails).

$$Kurt = \frac{\mathrm{E}\left[(\mathcal{Y} - \bar{\mathcal{Y}})^4\right]}{(\mathrm{E}[(\mathcal{Y} - \bar{\mathcal{Y}})^2])^2}$$

Surrogate Model's Learnability.

- *Misclassification Rate (MCR)*: We create a naive classifier $\mathcal{C}_n : X \to \mathcal{Y}$ that turns text, X, into predicted classes, $\mathcal{Y}$, and analyze the performance of this classifier because we think misclassification is related to its robustness. (Sun et al., 2020) shows that typos affect the robustness of the transformer-based model, so we choose the character-based CNN (Zhang et al., 2015) model for $\mathcal{C}_n$ because it can exploit character-level properties. Suppose classfier $\mathcal{C}_n$ turns a set T of text x_i with true class y_i into a predicted class $\hat{y}_i = \mathcal{C}_n(x_i)$. Misclassification Rate is expressed by the following formula:

$$MCR = \frac{|\{\hat{y}_i \neq y_i | (x_i, y_i) \in T\}|}{|T|}$$

Token-Based Statistics. We use a tokenizer $\mathcal{T}$, namely Bert-base-cased tokenizer (Devlin et al., 2018), to convert a sentence into an array of tokens. The notation X is a text set in the training dataset $\mathcal{S}_{train}$. Tokenizer $\mathcal{T}$ converts each text $x_i \in X$ into a list of M_i tokens $\{t_{ij}\}_{j=1}^{M_i}$, $\mathcal{T} : t_i \to \{t_{ij}\}_{j=1}^{N_i}$. Denote Y the collection of lists of tokens, so T turns X into Y. The formulas for those syntactic features are illustrated as followings:

$$\begin{aligned} \text{Avg. \# tokens} &= \frac{1}{|X|}\sum_{i=1}^{|X|} M_i \\ \text{\# unique tokens} &= |\{t | t \in \cup_{i=1}^{|X|} \{t_{ij}\}_{j=1}^{M_i} \\ &\quad \& \ t \text{ exists once}\}| \\ \text{Min \# tokens} &= min(\{M_i\}_{i=1}^{|X|}) \\ \text{Max \# tokens} &= max(\{M_i\}_{i=1}^{|X|}) \end{aligned} \tag{4}$$

In addition, the *total number of classes* is number of possible classes of dataset $\mathcal{D}$ from which the subdataset S_{train} is sampled

A.3 Evaluation Metrics

Denote $\hat{\mathcal{A}}$ and $\mathcal{A}$ the predicted ASR made by $\mathcal{P}$ and the actual ASR. The metrics we use to evaluate ASR predictors include the followings:

- *Root Mean Squared Error (RMSE)*: This is the square root of the mean square error (MSE), the average of the squared differences between the predicted values and the actual values.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\mathcal{A}_i - \hat{\mathcal{A}}_i\right)^2}{N}}, \quad (5)$$

where N is the number of predicted values.

- *Mean Absolute Error (MAE)*: This is the average of the absolute differences between the predicted values and the actual values. It is less sensitive to outliers than MSE, but may not penalize large errors as heavily.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|\mathcal{A}_i - \hat{\mathcal{A}}_i\right| \quad (6)$$

- *R-squared (R2)*: This is measured by the ratio between the mean squared error of a regression model and the variance of the target variable. It ranges from 0 to 1, with higher values indicating better performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\mathcal{A}_i - \hat{\mathcal{A}}_i)^2}{\sum_{i=1}^{N}(\mathcal{A}_i - \bar{\mathcal{A}})^2}, \quad (7)$$

where $\bar{\mathcal{A}} = \frac{\sum_{i=1}^{N}\mathcal{A}_i}{N}$

- *Mean Absolute Percentage Error (MAPE)*: This is the average of the absolute percentage differences between the predicted values and the actual values. It is commonly used in forecasting applications.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\mathcal{A}_i - \hat{\mathcal{A}}_i}{\mathcal{A}_i}\right| \quad (8)$$

- *Explained Variance Score (EVS)*: This is the proportion of variance in the target variable that is explained by the model relative to the total variance. It ranges from 0 to 1, with higher values indicating better performance.

$$EVS = 1 - \frac{Var(\mathcal{A} - \hat{\mathcal{A}})}{Var(\mathcal{A})} \quad (9)$$

A.4 Experiment Setup

Hardware Specifications. We use one GPU NVIDIA RTX A6000 and 32 CPUs AMD Ryzen Threadripper PRO 5975WX 32-Cores for our experiment.

A.5 Error Analysis

We examine how features impact error in ASR prediction. Initially, we employ the Random Forest model to predict ASR for test samples. Subsequently, test samples exhibiting errors surpassing the 70th percentile are categorized as False; otherwise, they are labeled as True. Then, logistic regression is applied to distinguish outlier test samples. Ultimately, the absolute magnitude of parameter weights from the logistic regression model is utilized to gauge feature significance. Essentially, these parameter weights indicate the degree of error influencing ASR prediction.

Sampling Datapoints.

The dataset list, $\mathcal{D}_l$, that we have used includes AG News, Amazon Review Full, Amazon Review Polarity, DPedia, Yahoo Answers, Yelp Review Full, Amazon Review Polarity, Banking 77, Emoji-TweetEval. In $\mathcal{D}_l$, we divided the datasets into two groups: $\mathcal{D}_a = \{$ AG News, Amazon Review Full, Amazon Review Polarity, DPedia, Yahoo Answers, Yelp Review Full, Amazon Review Polarity $\}$ and $\mathcal{D}_b = \{$ Banking 77, Emoji-TweetEval $\}$. The datasets in $\mathcal{D}_a$ have label counts of 2, 4, 5, 10, and 14 and are confined to a few settings, such as Yelp reviews, news articles, Yahoo inquiries, etc while those in $\mathcal{D}_b$ are 77 and 22. Due to the lack of diversity within each group of label counts, datasets in $\mathcal{D}_a$ are kept, whereas datasets in $\mathcal{D}_b$ are slightly adjusted to boost contextual variety for each group of label count. When a sub-dataset $(S^i_{\text{train}}, S^i_{\text{val}}, S^d_{\text{test}})$ in $\boldsymbol{Q}$ introduced in Algorithm 1 is sampled, if it is in the $\mathcal{D}_b$, the number of classes of that sub-dataset will also be randomly converted to 2, 4, 5, 10, or 14. For example, to convert a 22-label Emoji-TweetEval dataset into a 4-label dataset, the labels from 1 to 5, from 5 to 10, from 11 to 15, and from 16 to 20 will be converted into the new labels 1, 2, 3, 4, while samples with residual labels 21, 22 will be discarded.

After that, we sampled 500 data points of train data features and attack success rates in 72 hours.

BERT and ROBERTA hyperparameters. The tokenizer has a maximum length of 512 words. The learning rate, weight decay, and warmup step are 5e-4, 0.01, and 500, respectively. We train 5 epochs in that, from our observations, it is enough for the model to converge and get good inference results on the test dataset.

Character-level CNN. A tokenizer converts each character in a sentence into a one-hot vector in this model. It can only be 1024 characters long. There are six CNN layers, the kernel size of the first one and the five following are 3 and 7 respectively. The first and final CNN layer is followed by a 3 kernel size 1D pooling layer.

HDBSCAN. The minimum cluster size is five. The metric is Euclidean.

Gradient Boosting Regressor. The learning rate, maximum bin, and number of estimators are 0.05, 400, and 5000, respectively.

B Discussion

B.1 Confounding Factors

In this work, we assume two main factors (variables) that affect adversarial robustness: training data and a model's architecture. Most existing works analyze a model's adversarial robustness after it has been fine-tuned on the training data, thus mixing the two factors, making the analysis of how much training data (alone) correlates with adversarial robustness difficult. Thus, in this work, we separate the two factors and focus only on using training data as the input to measure the adversarial robustness for a specific model's architecture without fine-tuning such a model (fixed second variable–i.e., model architecture, and vary the first variable–i.e., training data).

Another possible confounding factor that affects both the training data and a model's adversarial robustness is the label and data curation process. For example, some malicious actors might intentionally poison the training data to affect the training features, labels, and a model's adversarial robustness. We assume that all of our training data is clean (we use official, published sources of all datasets) and we do not expect any confounding factors might affect our analysis.

There might be other confounding factors that we might overlook. Within the pioneering nature of our work on this research topic, we hope to see future works that explore those confounding factors, for example, from the perspective of causality.

B.2 Contextual Features

More nuanced, contextual, and semantic features of the training data will be useful for adversarial robustness prediction. In fact, our proposed framework already leverages the more nuanced aspects of fine-tuning data such as context and semantics by representing the original text using the Universal Sentence Encoder (USE) during the feature engineering step in Sec. 3.2. However, there might be other more complex features that would only be captured using complex neural network models, which then might inhibit interpretability and increase the runtime. For instance, if we aim to duplicate our training process but opt for an efficient neural network like CNN, we would have to train it with training examples. Each of them would consist of 900 sentences, where the average sentence length is 60 tokens. This results in an input size as large as 900×256 for each training example, with 256 as the word embedding size. This can be also considered as an image of 900×256 dimension, which is much larger compared to images of 28×28 dimension in the MNIST dataset. If we are not using CNN but a simple 1-hidden layer neural network with only 10 neuron units, we would need over 138M model parameters. We would expect a much longer runtime compared to our approach.

B.3 Relationship between compactness of input embedding and adversarial robustness

(Pal et al., 2023) introduces the theoretical concept of input data concentration, demonstrating that a robust classifier emerges when the input embedding is concentrated. On the other hand, (Si et al., 2021) employs augmentation techniques to enhance the density and concentration of the input embedding. Consequently, classifiers trained on such embeddings exhibit greater robustness. However, the approach of (Si et al., 2021) lacks an explicit explanation of the underlying rationale and fails to establish a direct correlation between input embedding concentration and adversarial robustness. Our research complements the findings of (Si et al., 2021) by revealing that sparser input embeddings lead to greater model robustness, while denser inputs result in decreased robustness as shown in Finding 3 in Section 6.

B.4 Generalization to comprehensive transformer architecture

We prove that our framework can be extended for all types of transformers including encoder-only (BERT, RoBERTa, ELECTRA), decoder-only (GPT2), and encoder-decoder (BART), shown in Table 3. The RMSE of interpolation ASR prediction is good for not only encoder-only transformers (range from 0.031 to 0.070) but also decoder-only and encoder-decoder transformers (0.025 and 0.028 respectively).

	Metric	Interpolation			Extrapolation		
		GB	**LR**	**RF**	**GB**	**LR**	**RF**
BERT	RMSE↓	0.059 ± 0.000	0.072 ± 0.000	**0.055 ± 0.000**	0.169 ± 0.005	0.063 ± 0.003	**0.063 ± 0.001**
	R^2↑	0.892 ± 0.006	0.841 ± 0.007	**0.904 ± 0.005**	0.394 ± 0.177	0.871 ± 0.122	**0.885 ± 0.033**
	MAE↓	0.040 ± 0.000	0.053 ± 0.000	**0.037 ± 0.000**	0.128 ± 0.006	**0.040 ± 0.001**	0.045 ± 0.000
	EVS↑	0.895 ± 0.006	0.846 ± 0.007	**0.907 ± 0.005**	0.522 ± 0.089	0.892 ± 0.060	**0.908 ± 0.021**
	MAPE↓	0.077 ± 0.001	0.101 ± 0.001	**0.071 ± 0.000**	0.278 ± 0.020	**0.086 ± 0.006**	0.102 ± 0.004
RoBERTa	RMSE↓	0.037 ± 0.000	0.056 ± 0.000	**0.031 ± 0.000**	0.206 ± 0.005	0.073 ± 0.003	**0.061 ± 0.001**
	R^2↑	0.959 ± 0.000	0.907 ± 0.001	**0.972 ± 0.000**	0.139 ± 0.145	0.829 ± 0.205	**0.900 ± 0.019**
	MAE↓	0.028 ± 0.000	0.044 ± 0.000	**0.025 ± 0.000**	0.176 ± 0.006	**0.042 ± 0.001**	0.044 ± 0.000
	EVS↑	0.961 ± 0.000	0.911 ± 0.001	**0.972 ± 0.000**	0.309 ± 0.109	0.846 ± 0.153	**0.922 ± 0.010**
	MAPE↓	0.054 ± 0.000	0.083 ± 0.000	**0.048 ± 0.000**	0.385 ± 0.032	**0.083 ± 0.003**	0.095 ± 0.004
ELECTRA	RMSE↓	0.107 ± 0.001	0.084 ± 0.002	**0.070 ± 0.001**	0.135 ± 0.004	0.148 ± 0.009	**0.073 ± 0.000**
	R^2↑	0.411 ± 0.492	0.635 ± 0.194	**0.686 ± 0.490**	0.450 ± 0.240	0.348 ± 0.694	**0.864 ± 0.007**
	MAE↓	0.083 ± 0.001	0.057 ± 0.001	**0.047 ± 0.000**	0.100 ± 0.005	0.064 ± 0.000	**0.039 ± 0.000**
	EVS↑	0.505 ± 0.293	0.677 ± 0.152	**0.729 ± 0.326**	0.513 ± 0.174	0.361 ± 0.671	**0.870 ± 0.005**
	MAPE↓	0.151 ± 0.006	0.105 ± 0.004	**0.084 ± 0.003**	0.180 ± 0.012	0.129 ± 0.002	**0.077 ± 0.000**
GPT2	RMSE↓	0.093 ± 0.002	0.026 ± 0.000	**0.025 ± 0.000**	0.110 ± 0.002	0.147 ± 0.009	**0.078 ± 0.000**
	R^2↑	−0.468 ± 37.303	0.888 ± 0.105	**0.890 ± 0.106**	0.523 ± 0.135	−0.013 ± 3.437	**0.794 ± 0.005**
	MAE↓	0.067 ± 0.001	**0.020 ± 0.000**	0.022 ± 0.000	0.079 ± 0.002	0.069 ± 0.000	**0.051 ± 0.000**
	EVS↑	0.019 ± 10.871	0.911 ± 0.056	**0.913 ± 0.049**	0.545 ± 0.122	0.005 ± 3.314	**0.801 ± 0.005**
	MAPE↓	0.107 ± 0.004	**0.028 ± 0.000**	0.030 ± 0.000	0.136 ± 0.005	0.126 ± 0.001	**0.009 ± 0.000**
BART	RMSE↓	0.052 ± 0.000	0.041 ± 0.001	**0.028 ± 0.000**	0.107 ± 0.003	0.070 ± 0.003	**0.068 ± 0.001**
	R^2↑	0.856 ± 0.014	0.885 ± 0.028	**0.995 ± 0.001**	0.423 ± 0.264	0.743 ± 0.222	**0.813 ± 0.019**
	MAE↓	0.039 ± 0.000	0.028 ± 0.000	**0.022 ± 0.000**	0.074 ± 0.003	**0.032 ± 0.000**	0.036 ± 0.000
	EVS↑	0.875 ± 0.009	0.896 ± 0.044	**0.960 ± 0.001**	0.501 ± 0.124	0.747 ± 0.213	**0.822 ± 0.017**
	MAPE↓	0.070 ± 0.002	0.053 ± 0.002	**0.036 ± 0.000**	0.124 ± 0.005	**0.063 ± 0.001**	0.076 ± 0.001

Table 3: ASR results (mean±std) under interpolation and extrapolation prediction on BERT, RoBERTa, ELECTRA, BART and GPT2 using three classifiers, namely Gradient Boosting (GB), Linear Regression (LR), and Random Forest (RF).