Attack-prone Features

Anonymous ACL submission

Abstract

Recently, BERT and ROBERTA models have been widely used for problems in the field of natural language processing. Besides the effectiveness of those models, robustness and reliability are issues that attract huge attention. People often use a pre-trained model and fine-tune it for a variety of tasks. Therefore, if we fix machine learning models, the robustness and reliability of these models vary depending on the training dataset. So what factors of train data affect the robustness of the model and what is the order of their influence? Since the NLP model is fixed and the train data is variable, is it possible to directly predict the robustness of the NLP model based on the train data? In this paper, we focus on addressing these two questions. Specifically, we try to show how embedding space, distribution of labels in train data, and syntactic structures affect the robustness of the model. In addition, we use Gradient Boosting and Linear Regression to predict the robustness of the NLP model based on the features just mentioned. The implementation for our experiments is available at this Gihub page.

1 Introduction

Recently, with the advancement of pre-trained natural language processing models such as BERT (Devlin et al., 2018a), Roberta (Liu et al., 2019) or XLNet (Yang et al., 2019), etc., the problems in the field of natural language processing have been solved with high performance. However, that is not enough when the models are deployed in areas that require the robustness and reliability of the model such as banking and law (Rodríguez Cardona et al., 2021; Sanz-Urquijo et al., 2022; Ashley, 2019). Several studies have shown that, despite the high performance, the above models are not robust to adversarial examples (Goodfellow et al., 2014; Papernot et al., 2016a; Ilyas et al., 2019). Therefore, improving and explaining the lack of robustness of models is a very important research direction and increasingly attracts the attention of many researchers.

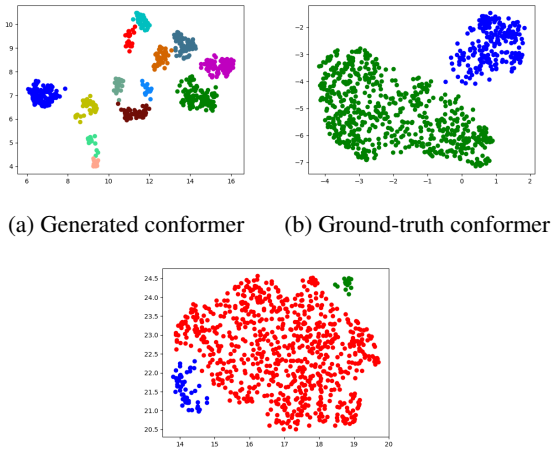
A study (Mao et al., 2022) of Vision Transformers (ViTs) models found that robustness depends on learning the low-order characteristics of the data, location encoding, meticulous multi-layer design, the number of self-attention head, replacing CLS with global average pooling on output tokens. Since then, the author has improved robust components to create a new transformer called Robust Vision Transformer (RVT). Another study (Zhang et al., 2022b) explains that the robustness of the model is affected by the model's ability to learn with the perturbation of the data, the model that learns the perturbation better is less robust. (Ilyas et al., 2019; Tsipras et al., 2018) show that the robustness of the vision model can be attributed to non-robust features in the representation space that are sensitive to adversarial examples. Inspired by the above studies, (Zhang et al., 2022a) proposed to use an information bottleneck layer between the output layer of BERT and the text classifier to keep the task-specific features. (Han et al., 2022) suggests that the robustness of the model is affected by contaminated samples. Therefore, they proposed robust kernel density estimation (RKDE) in the self-attention mechanism by reducing the weight of bad samples during estimation.

The above methods have explained the influence of the sample on the robustness of the model and improved the robustness through changes in the underlying architecture. On the other hand, people often load a pre-trained model and fine-tune them to solve different tasks. So what people can tweak to make the model more robust is the dataset used for training the model (besides the model's hyperparameters). To the best of our knowledge, there is still no research that deeply and comprehensively explains the properties of the training data that affect the robustness of NLP models, especially modern transformer-based models. If the

NLP model is fixed and the training data is variable, one question arises, can we directly predict the robustness of the NLP model just based on the model training dataset? In this study, we focused on addressing the following questions:

- **Q1:** What features of the training data set affect the robustness of an NLP model?
- **Q2:** Is it possible to build a model to predict the robustness of an AI model based on the training data set alone?

We test hypothesis **H1** to answer the question **Q1** that the main factors affecting the robustness of the model lie in the embedding space (data with the same label are distributed close or far apart and how many clusters are in the embedding space, etc) illustrated in Figure 1, label distribution depicted in Figure 3 and syntactic structure of the training data demonstrated in Figure 2. We also prove the hypothesis **H2** an answer to question **Q2** that it is possible to predict the robustness of the model based on the training data by designing a predictor $\mathcal{P}$ that can predict the robustness of the NLP model $\mathcal{M}$ given the training data features $\mathcal{X}$ used in **H1**. Specifically, we use the Gradient Boosting and Linear Regression models for predicting the robustness of the model and ranking the influence of features $\mathcal{X}$ on its prediction. Finally, we will discuss in more depth the useful applications of predictor $\mathcal{P}$ in Section 5.

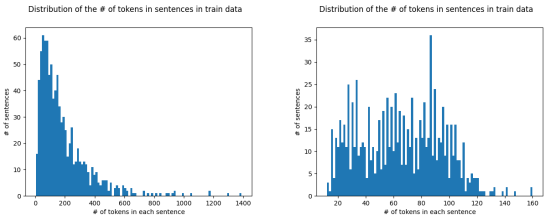


(a) Generated conformer (b) Ground-truth conformer

(c) Overlap of the ground-truth

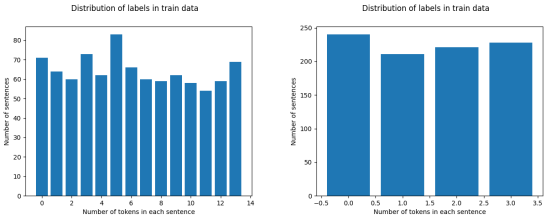
Figure 1: Conformer of

Within the scope of this paper, we focus on studying the sustainability of two modern and popular



(a) Generated conformer (b) Ground-truth conformer

Figure 2: Conformer of



(a) Generated conformer (b) Ground-truth conformer

Figure 3: Conformer of

models, including BERT and ROBERTA. In summary, our main contributions include:

- Find out features of the training dataset that affect the robustness of BERT and ROBERTA model
- Rank the influence of features of training dataset on the robustness of BERT and ROBERTA model
- The first paper introducing a machine learning model to predict the attack success rate of BERT and ROBERTA model just based on a training dataset

2 Related Work

Adversarial attacks for NLP.
Natural Language Processing (NLP) has grown in popularity in recent years as the amount of textual data available has increased. But, as technology has advanced, so have security threats, such as adversarial attacks. The malicious manipulation of input data to fool or influence the output of machine learning models is known as an adversarial attack.

The general pattern for adversarial attacks is to first select and rank the most important words affecting the certainty of the classifier and then gradually modify them so that the classifier's certainty of the target class diminishes until the attack succeeds. Therefore, the main components of an adversarial attack usually include the importance of selection

and transform mechanism. There are two settings, black-box, and white-box. In the white-box setting, the method of choosing important words is easier, since the gradients of the model can be checked, the change of words that make the gradients change more is more important. However, in the black-box context, model gradients are not available, so a metric is required to assess the importance of each word in the sentence.

First, in the black-box context, (Alzantot et al., 2018) presents a word-level strategy that attempts to generate semantically and syntactically similar adversarial examples through genetic algorithms. They first employ a perturbing approach to generate multiple candidates and then apply the genetic algorithm until achieving a desired example. Bert attack (Li et al., 2020) generates a list of the most important words. Then BERT (Devlin et al., 2018a) is used to generate top-k replacement words for each word in that list. Next, the replacement words are alternately substituted in order of decreasing perplexity. DeepWordBug (Gao et al., 2018) is an adversarial attack method at the character level. This method proposes a Combination Score of the Temporal Tail Score and Temporal Score, as well as a way for slightly changing words by regulating the edit distance from the original sample. The words are modified in turn by substituting a random letter for a letter in the word, deleting a random letter from the word or adding a random letter in the word, and exchanging two adjacent letters in the word. PWWS (Ren et al., 2019) is a word-level approach. It identifies the word in the sentence with the largest variation for each one. The replacement order is determined by the value of the score function, which is the probability weighted by word saliency. TextFooler (Jin et al., 2020) is a method that first ranks the words by their importance scores, and then find replacements having the same meaning and part-of-speech for each word in the sentence until the prediction of the target model is altered.

In terms of the white-box, (Papernot et al., 2016b) searches the dictionary for the replacement word z whose difference from the original input word x is closest to the derivative of the loss function relative to x. (Samanta and Mehta, 2017) remove the adverb, insert it before the adj, or replace the word with the same POS tag. (Ebrahimi et al., 2017) is a character-level that allows for the addition and removal of characters in order to maximize loss-increasing direction.

Interpreting the adversarial robustness of models.

In this part, we list some studies that explain the factors affecting the robustness of models. (Zhao et al., 2022) takes a causal approach and believes that the non-robust model is caused by the influence of confounders and they proposed Causal Intervention by Semantic Smoothing (CISS) to eliminate the influence of such confounders and increase the robustness of the model. According to (Zhang et al., 2022a), the non-robust text classification model is caused by its non-robust features. As a result, they improved the model by including a bottleneck layer to eliminate the effects of those features. (Han et al., 2022) attributed the non-robust Transformer model to outliers, and they presented a resilient Transformer framework called Transformer-RKDE based on replacing the dot-product attention by attention arising from the robust kernel density estimators (RKDE) associated with the robust kernel regression problem.

However, the studies described above merely make hypotheses without thoroughly examining the impact of the factors they propose. The following studies hypothesize which factors influence the robustness of the model and provide concrete empirical evidence. (Jia and Liang, 2017) addresses the question of whether reading comprehension systems can correctly interpret language. They presented an adversarial approach of adding sentences without changing the answer or misleading people. After several experiments, they discovered that adversarial perturbation had a significant impact on the accuracy of the system. The cause of influence is expected that the model will be robust to adversarial perturbation when the question has precisely n-gram matches with the original text. In addition, the model is less robust for short questions. (Yu et al., 2018) explains the relationship between adversarial robustness, parameter space, and input space. To do that, they propose a definition of the decision plane to visualize the trajectories of the adversarial attack on the decision plane in the input space. By doing so, they show that many adversarial attacks obey the geometry of the decision plane where the decision boundary lies within the smallest distance. Finally, they formalize the adversarial robustness indicator by relating the geometric properties of the Jacobian and Hessian eigenvalues. According to Study (Sun et al.,

2020), the BERT language model is not robust to typographical errors including: insert, delete, swap, and mistype. After intensive experiments, they showed that spelling errors in different words of a sentence do not affect the same. Spelling errors in informational words make the damage more severe. Mistype is the most damaging factor, compared to insertions, deletions, etc. Humans and machines have different focuses on recognizing adversarial attacks.

Inspired by the above studies, we propose factors affecting the robustness of models including the embedding of input text, distribution of labels in training data, and syntactic features.

Accumulated Local Effects. The ALE plot shows the effect of one predictor variable on the change in the response of the model when the value of the predictor variable changes without being affected by the interaction of the other predictor variables.

Assuming X_j is the predictor variable we want to see its effect on the outcomes of model f and $X_{\setminus j}$ are the remaining predictor variables. We have an obvious equation.

$$f(X_j = x_j, X_{\setminus j}) = \int_{z=min(X_j)}^{x_j} \frac{\partial f(X_j, X_{\setminus j})}{\partial X_j} dz + f(min(X_j), X_{\setminus j})$$

The ALE function shows the effect of X_j on the outcome of f by removing the effect of X_j through calculating expectation over $X_{\setminus j}$.

$$f_{j,ALE}(x_j) \equiv E_{X_{\setminus j}}\left[f(X_j = x_j, X_{\setminus j})\right] = \int_{z=min(X_j)}^{x_j} E_{X_{\setminus j}}\left[\frac{\partial f(X_j, X_{\setminus j})}{\partial X_j} | X_j = z\right] dz + C,$$

where C is a constant and $C = E_{X_{\setminus j}}[f(min(X_j), X_{\setminus j})]$.

Normally the function f is nondifferentiable and $X_{\setminus j}$ is a multi-dimension variable, the derivative of f is replaced by the limit of finite differences in $f_{j,ALE}(x_j)$ and its expectation is calculated by sampling (Apley and Zhu, 2020).

3 Method

Our goal is to create a table that includes the features of the training datasets and the model's success rate when trained on those datasets. Specifically, refer to the table... To diversify the number of training datasets and minimize computational resources while ensuring fairness, we sample sub-datasets from large, well-established datasets. Then, based on the features extracted from sub-datasets, we can predict the success rate of attacks using regression algorithms for tabular data and then analyze the influence of those features on the robustness of the model. Our method has five phases, as illustrated in Figure 4:

- Prepare train/val/test data
- Extract features from training data
- Predict Adversarial Robustness
- Train a robustness predictor
- Analyze and rank the influence of features

3.1 Prepare train/val/test data

This phase including steps 1 and 2 in Figure 4 is done to prepare for the next two phases which include extracting features from train data (phase 4) and predicting the adversarial robustness of the inspected model (phase 5). The goal of this phase is to generate a large number of datasets that can simulate all datasets in the wild. To diversify the number of datasets and minimize computational resources while ensuring fairness, we sample those from a list of large well-established text classification datasets, denoted $\mathcal{D}_l$, including AG News, Amazon Reviews, DBpedia, IMDb, Sogou News, Yahoo Answers, and Yelp Reviews.

In step 1, we sample uniformly a dataset $\mathcal{D} \in \mathcal{D}_l$, $\mathcal{D}$ comprised of two separate parts test dataset $\mathcal{D}_{test}$ and train dataset $\mathcal{D}_{train}$. In step 2, we randomly take 100 and 900 sentences in the set $\mathcal{D}_{test}$ and $\mathcal{D}_{train}$ respectively to generate validation data $\mathcal{S}_{val}$ and train data S_{train}. In addition, 100 sentences are randomly taken from $\mathcal{D}_{test}$ to test data. The above procedure is repeated many times to generate sets of 3 train/val/test ($\mathcal{S}_{train}, \mathcal{S}_{val}, \mathcal{S}_{test}$). To ensure that the classifier in the training and evaluation step is efficient and fair, each dataset in the triple ($\mathcal{S}_{train}, \mathcal{S}_{val}, \mathcal{S}_{test}$) do not intersect each other.

Test data, $\mathcal{S}_{test}$, are used to predict attack success rate(ASR), also used as indicator for the model robustness, in step 3. Therefore, to ensure fairness for sets $\mathcal{S}_{train}, \mathcal{S}_{val}, \mathcal{S}_{test}$ taken from the same dataset $\mathcal{D}$, test datasets $\mathcal{S}_{test}$ taken from the same large dataset $\mathcal{D}$ are the same. The illustration for split strategy is depicted in Figure 5.

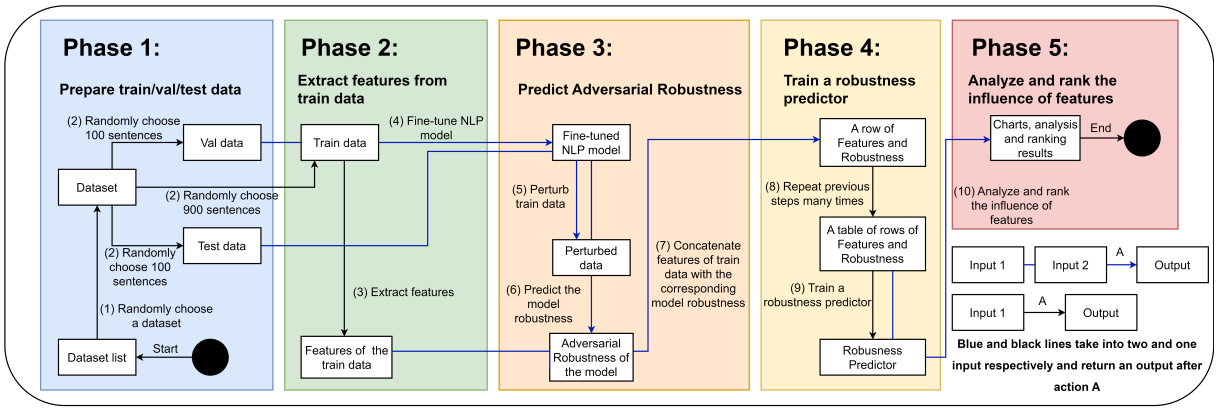


Figure 4: Flow chart of our experiment has five phases. The black and blue arrow takes one and two objects as input respectively and returns an output.

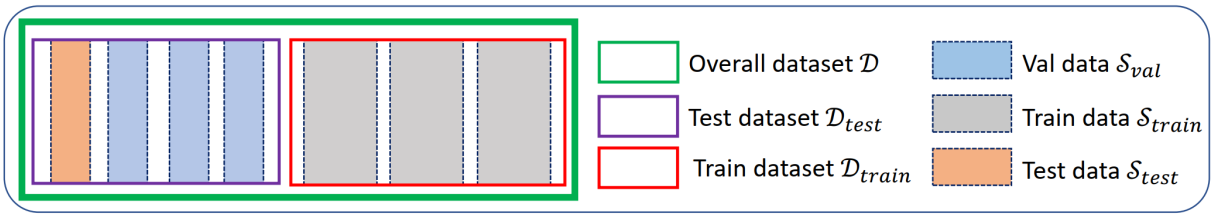


Figure 5: This figure shows the sample and split strategy. The entire well-established dataset is represented by a green rectangle. The entire dataset including the train and test datasets represented by red and purple rectangles, respectively. Each pair of gray and blue rectangles represents the sampled Train and Val datasets, respectively. Each well-establish dataset has only one sampled Test dataset represented by an orange rectangle. The sampled Train/Val/Test datasets do not intersect and are represented by dotted rectangles.

3.2 Extract features from training data

This phase includes step 3 in Figure 4 extracting the characteristic features of the train data to analyze their influence on the robustness of the model in phase 5. Features are divided into 3 main groups: Embedding, Distribution of labels, and Syntactic Features. In Embedding, we split into 2 smaller groups including Class Separation and Clustering. Within each group/subgroup, we use a variety of quantitative indicators to describe the characteristics and properties of that group. The groups and indicators are illustrated in Figure 6.

Embedding. Inspired by (Yu et al., 2018) which studies the influence of input space on model robustness, we also propose several indicators that can quantitatively describe input space, including the mean distance between classes, Fisher's discriminant ratio, Calinski-Harabasz Index, Davies-Bouldin Index, the number of clusters. 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, D and E are used to measure the density or sparseness of the embedding space. For mapping input text into multidimensional space, we use a pre-trained Universal Sentence Encoder (Cer et al., 2018).

■ For indicators for class separation, assume that the input text is spatially mapped vector $\mathcal{X} = \{x_i\}_{i=1}^{N}$ and $\mathcal{Y} = \{y_i\}_{i=1}^{N}$ are their labels. Vectors with the same label will be classified into the same clusters. $\mathcal{C} = \{C_i, N_i, m_i\}_{i=1}^{K}$ are the clusters in the embedding space and the number of vectors in that cluster, and the center of that cluster respectively, where K is the number of possible labels.

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

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

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

Let denote r_i, r, d_{ij} the average distance between each point of cluster i and the centroid of that cluster also known as cluster diameter, the global di-

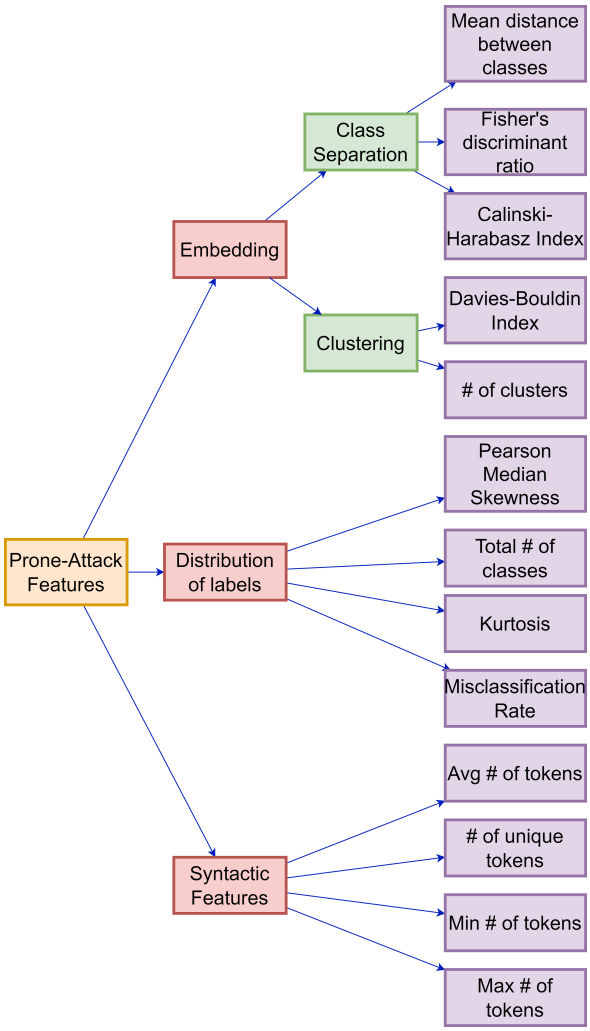


Figure 6: Taxanomy of Features that have influences on the robustness of models. They are divided into three big groups, including Embedding, Distribution of labels, and Syntactic Features located in the red boxes. Embedding is further subdivided into two groups Class Separation and Clustering in green boxes. Indicators are in the blue box. Indicators are in purple boxes.

ameter and the distance between cluster centroids i and j.

$$r_i = \frac{1}{N_i} \sum_{i \in C_i} (x_i - m_i)(x_i - m_i)^T$$
$$r = \frac{1}{N} \sum_{i \in C} (x_i - m)(x_i - m)^T$$
$$d_{ij} = (m_i - m_j)(m_i - m_j)^T$$

The formulas for the Mean Distance between classes, Fisher's Discriminant Ratio, 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.

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

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.

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

■ 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 based on their labels. Hence, $\mathcal{C} = \{C_i, N_i, m_i\}_{i=1}^{K}$ are the clusters in the embedding space and the number of vectors in that cluster, and the center of that cluster respectively, where K 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:

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

where

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

Distribution of labels. Skewed training data can lead to biased predictions since the model tends to learn from the majority class, and may fail to recognize patterns in the minority class. It also leads to poor generalization. A model trained on skewed data may not generalize well to new data that has a different distribution. This is because the model has learned to overemphasize certain features or patterns that are specific to the skewed data. Therefore, we propose several indicators to quantify the skewness and properties of the distribution of input text. Let denote $\mathcal{Y}$ a random variable representing for possible labels in the train dataset $\mathcal{S}_{train}$.

- *Pearson Median Skewness (SMK)*: 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.

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

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

- *Total number of classes*: number of possible class of dataset $\mathcal{D}$ from which the sub-dataset S_{train} is sampled

- *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).

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

- *Misclassification Rate (MCR)*: We create a naive classifier $\mathcal{C}_n : X \rightarrow \mathcal{Y}$ 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 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:

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

Syntactic Features. (Jia and Liang, 2017; Sun et al., 2020) show that the length of input text, typos affects the robustness of the model. Therefore, we examine the influence of some syntactic features on the robustness of the model. We use a tokenizer $\mathcal{T}$, namely Bert-base-cased tokenizer (Devlin et al., 2018b), to convert a sentence into an array of tokens. The notation X is a text set in the train dataset $\mathcal{S}_{train}$. Tokenizer $\mathcal{T}$ converts each text $x_i \in X$ into a list of M_i tokens $\{t_{ij}\}_{j=1}^{M_i}$, $\mathcal{T} : t_i \rightarrow \{t_{ij}\}_{j=1}^{N_i}$. Denote Y the collection of lists of tokens, so T turns X into Y. The formulas for those syntactic features are illustrated as followings:

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

3.3 Predict Adversarial Robustness

This phase includes steps 4, 5, and 6 in Figure 4. After extracting the features of the training data $\mathcal{S}_{train}$, we train a classification model $\mathcal{M}$ to measure the success rate of this model. Since the sampled datasets, $(\mathcal{S}_{train}, \mathcal{S}_{val}, \mathcal{S}_{test})$, have a different number of labels, we design a mutable architecture of classification head to fit each sampled dataset. In fact, we use BERT and ROBERTA implemented by the Huggingface(Wolf et al., 2019) library for classifier $\mathcal{M}$. In step 4, Val and Train dataset, $\mathcal{S}_{train}$ and $\mathcal{S}_{val}$, are used to fine-tune the model. To predict ASR objectively in steps 5 and 6, we use four methods to jam and attack the model, including one character-level method, DeepWordBug (Gao et al., 2018), three word-level methods method BERT (Li et al., 2020), PWWS (Ren et al., 2019), TextFoler (Jin et al., 2020) and then average them. The average ASR is also used as the robustness indicator for an NLP model.

3.4 Train a robustness predictor

This phase includes steps 8 and 9 in Figure 4. After performing steps 1 to 7 we get a datapoint(row) containing the characteristics of train data $\mathcal{S}_{train}$ and the attack success rate $\mathcal{A}$ of the model trained on $\mathcal{S}_{train}$. In step 8, we do steps from 1 to 7 many times to get more datapoints. Then we create a predictor $\mathcal{P}$ to predict the attack success rate $\mathcal{A}$ based on the features $\mathcal{F}$ of the train data, $\mathcal{P} : \mathcal{F} \rightarrow \mathcal{A}$. (Grinsztajn et al., 2022) shows that tree-based models still outperform deep learning on tabular data even though deep learning has made huge advances for learning on structured data such as image, language, and audio. Therefore, we choose 2 models that are simple, and fast but highly accurate in learning tabular data. In addition, that model will help with the future work covered in Section 5. Denote $\hat{\mathcal{A}}$ the predicted ASR, $\hat{\mathcal{A}} = \mathcal{P}(\mathcal{F})$. The metrics we use to evaluate the model include the followings:

- *Root Mean Squared Error (RMSE)*: This is the square root of the MSE, the average of the squared differences between the predicted values and the actual values. It is easier to interpret because it is in the same units as the target variable.

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

  , 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.

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

- *R-squared (R2)*: This is the proportion of variance in the target variable that is explained by the model. It ranges from 0 to 1, with higher values indicating better performance.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\mathcal{A}_i - \hat{\mathcal{A}}_i)^2}{\sum_{i=1}^{N}(\mathcal{A}_i - \bar{\mathcal{A}})^2} \quad (3)$$

  , 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.

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{\mathcal{A}_i - \hat{\mathcal{A}}_i}{\mathcal{A}_i}\right| \quad (4)$$

- *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.

$$EVS = 1 - \frac{Var(\mathcal{A} - \hat{\mathcal{A}})}{Var(\mathcal{A})} \quad (5)$$

3.5 Analyze and rank the influence of features

We provide two methods to analyze the influence of features including Accumulated Local Effects (ALE) Plot (Apley and Zhu, 2020) and linear regression.

4 Experiments & Results

BERT and ROBERTA hyperparameters. The tokenizer has a maximum length of 512 words. The learning rate , weight decay , and warmup steps is 5e-4, 0.01, and 500, respectively. We train 5 epochs.

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 of and five followings is 3 and 7 respectively. The first and final CNN layer is followed by a 3 kernel size 1D pooling layer.

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

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

Sampling Result. We sampling 300 datapoints. Table 1 shows 10 data points including features of the train data $\mathcal{S}_{train}$ and the attack success rate on BERT model trained on $\mathcal{S}_{train}$.

ASR Prediction After having the full We randomly split 60%, 20%, 20% for training, testing and scoring. Because the number of training data is low, the train and evaluation procedure is repeated 40 times and the final result is then averaged. Table 2 show the average $RMSE$, R^2, MAE, EVS, and $MAPE$ are 0.04, 0.93, 0.02, 0.94, and 0.05 respectively.

Analyze and rank influence.

Figure 7 and 7 shows the influence of continuous and discrete features on the robustness of BERT. Table 3 show the summary of trends, rankings, influence of features on the robustness of BERT.

5 Future Work

This

Limitations

Ethics Statement

Acknowledgements

References

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

Daniel W Apley and Jingyu Zhu. 2020. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086.

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.

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

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. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*.

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*.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. 2019. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*.

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. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

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*.

Avg # tokens	# unique tokens	Min # tokens	Max # tokens	MD	FDR	CHI	DBI	# clusters	PMS	# labels	Kurt	MCR	ASR-TF	ASR-PWWS	ASR-BERT	ASR-DWB
46.41	7425	14	161	0.44	1.99	594.83	2.43	6	0.39	4	1.52	0.75	0.6	0.58	0.57	0.2
97.99	8984	17	283	0.35	2.09	1878.43	2.59	2	0	2	1.00	0.61	0.78	0.82	0.72	0.24
192.59	9222	5	1372	0.456	1.47	1322.58	1.84	2	0	2	1.00	0.83	0.74	0.74	0.87	0.26
101.38	9026	16	265	0.36	2.05	1845.34	2.33	4	0	2	1.00	0.69	0.61	0.54	0.58	0.27
55.01	6036	4	396	0.34	2.57	253.81	2.63	2	-0.17	10	2.24	0.9	0.91	0.91	0.9	0.56
66.22	9675	11	219	0.61	1.37	93.57	1.77	14	0.10	14	2.78	0.91	0.45	0.37	0.62	0.09
102.12	9273	21	373	0.35	2.08	1863.77	2.40	2	0	2	1.00	0.79	0.7	0.7	0.84	0.35
98.67	8930	19	268	0.36	2.06	1851.17	2.38	4	0	2	1.00	0.55	0.72	0.71	0.91	0.32
103.35	9061	15	284	0.26	2.04	456.17	2.23	2	0.29	5	2.35	0.86	0.86	0.97	0.97	0.66
46.31	7491	16	205	0.42	2.02	602.01	2.39	5	-0.19	4	1.65	0.99	0.62	0.58	0.56	0.22

Table 1: Features of train datasets $\mathcal{S}_{train}$ and attack success rate of BERT model trained on $\mathcal{S}_{train}$.

RMSE	R2	MAE	EVS	MAPE
0.04	0.93	0.02	0.94	0.05

Table 2: The average result of ASR prediction after 40 times of training and evaluating.

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.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016a. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016b. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pages 49–54. IEEE.

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.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

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.

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*.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2018. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

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.

Haiteng Zhao, Chang Ma, Xinshuai Dong, Anh Tuan Luu, Zhi-Hong Deng, and Hanwang Zhang. 2022. Certified robustness against natural language attacks by causal intervention. In *International Conference on Machine Learning*, pages 26958–26970. PMLR.

A Example Appendix

This is a section in the appendix.

	# classes	# clusters	# unique tokens	Max # tokens	Min # tokens	Avg. # tokens	CHI	FR	KTS	MD	MR	PM	DBI
Influence range	0.031	0.096	0.020	0.023	0	0.1143	0.0066	0.0136	0	0.1781	0.0062	0.0022	0
Ranking	4	3	6	5	11	2	8	7	11	1	9	10	11
Proportional to ASR						✓					✓		
Inversely Propotional to ASR		✓	✓	✓						✓		✓	
Unchanged	✓				✓				✓				✓
Fluctuating							✓	✓					

Table 3: Table show the summary of trends, rankings, influence of features on the robustness of BERT

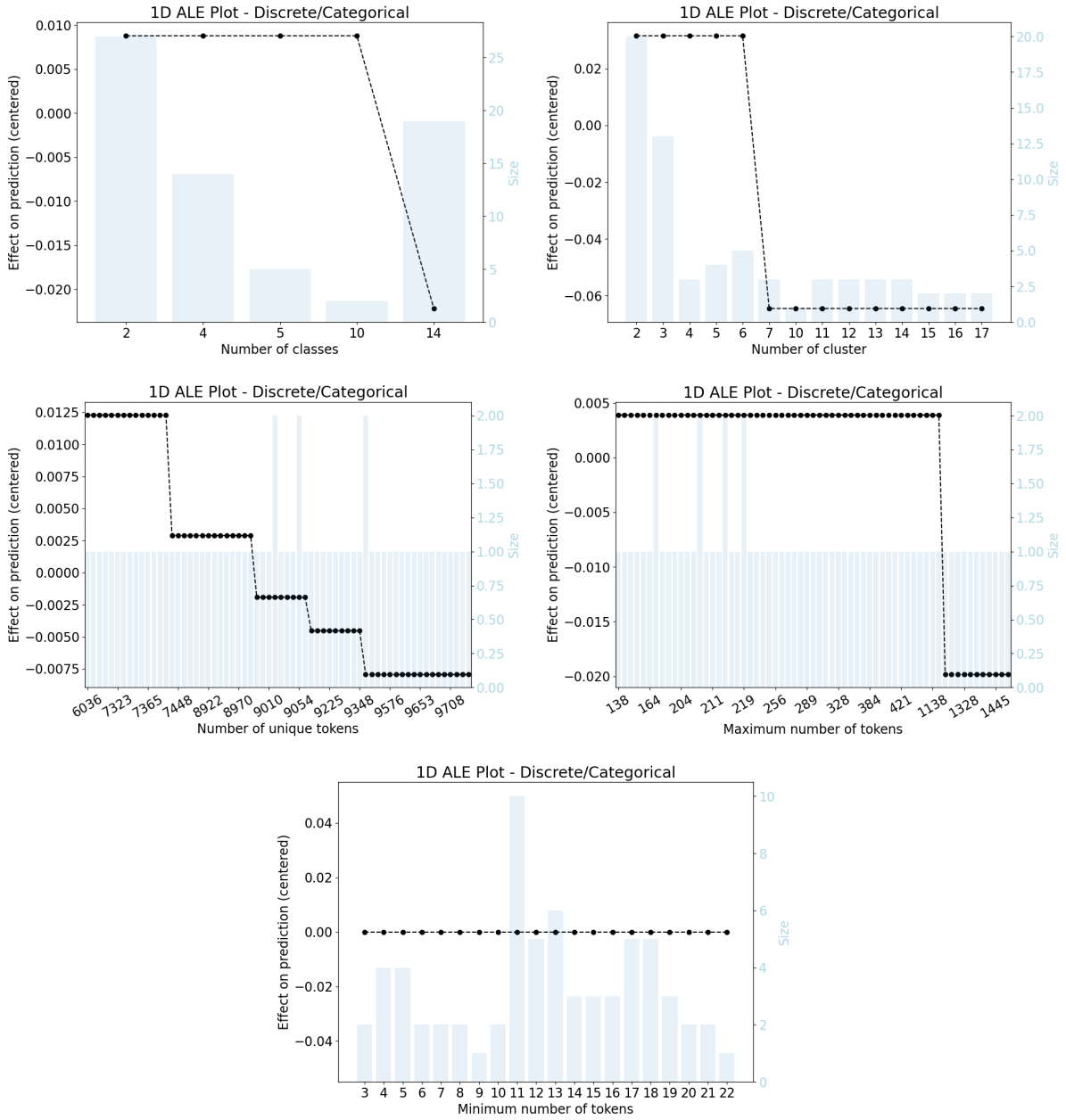


Figure 7: The graph shows the influence of discrete features on the robustness of BERT

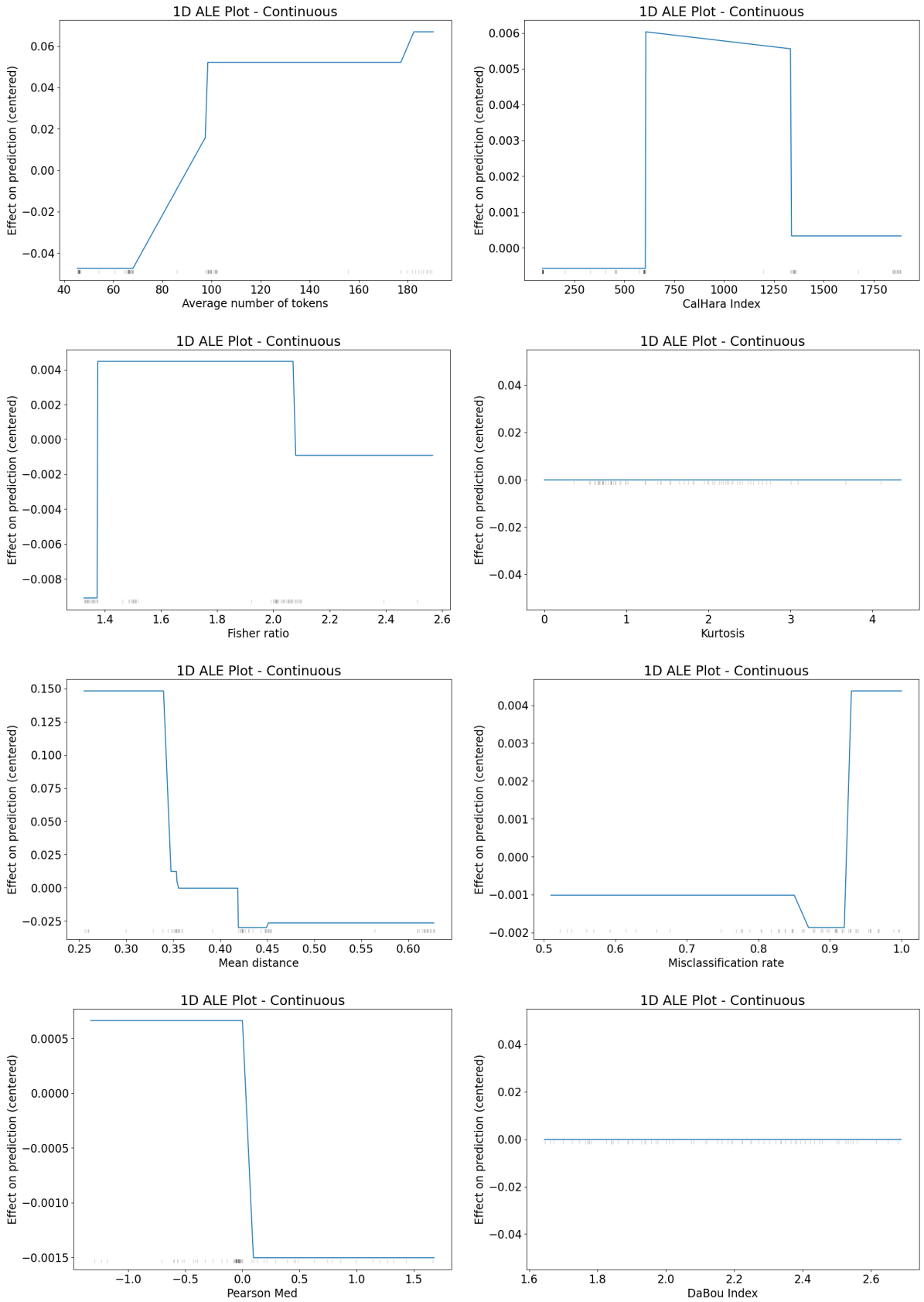


Figure 7: The graph shows the influence of continuous features on the robustness of BERT