What Models Know About Their Attackers: Deriving Attacker Information From Latent Representations

Zhouhang Xie[1] **Jonathan Brophy**[2] **Adam Noack**[2] **Wencong You**[2] **Kalyani Asthana**[1]
Carter Perkins[2] **Sabrina Reis**[2] **Zayd Hammoudeh**[2] **Daniel Lowd**[2] **Sameer Singh**[1]
[1] University of California, Irvine CA
[2] University of Oregon, Eugene OR
{zhouhanx, sameer}@uci.edu

Abstract

Adversarial attacks curated against NLP models are increasingly becoming practical threats. Although various methods have been developed to detect adversarial attacks, securing learning-based NLP systems in practice would require more than identifying and evading perturbed instances. To address these issues, we propose a new set of adversary identification tasks, *Attacker Attribute Classification via Textual Analysis (AACTA)*, that attempts to obtain more detailed information about the attackers from adversarial texts. Specifically, given a piece of adversarial text, we hope to accomplish tasks such as localizing perturbed tokens, identifying the attacker's access level to the target model, determining the evasion mechanism imposed, and specifying the perturbation type employed by the attacking algorithm. Our contributions are as follows: we formalize the task of classifying attacker attributes, and create a benchmark on various target models from sentiment classification and abuse detection domains. We show that signals from BERT models and target models can be used to train classifiers that reveal the properties of the attacking algorithms. We demonstrate that adversarial attacks leave interpretable traces in the feature space of both of pre-trained language models and target models, making AACTA a promising direction towards more trustworthy NLP systems.

1 Introduction

Accompanying the success of deep learning in Natural Language Processing, there has been a prevalence of adversarial attacks in text. Attackers modify input sequences to NLP models such as text classifiers and machine translation models slightly so that the prediction of the target model will deviate from its original output (Zhang et al., 2020). These attacks are known to be generalizable across models (Yuan et al., 2021), and can hurt the performance of NLP models that are used in real-world tasks, such as commercial sentiment classification APIs (Garg and Ramakrishnan, 2020) and Google Translate (Wallace et al., 2020).

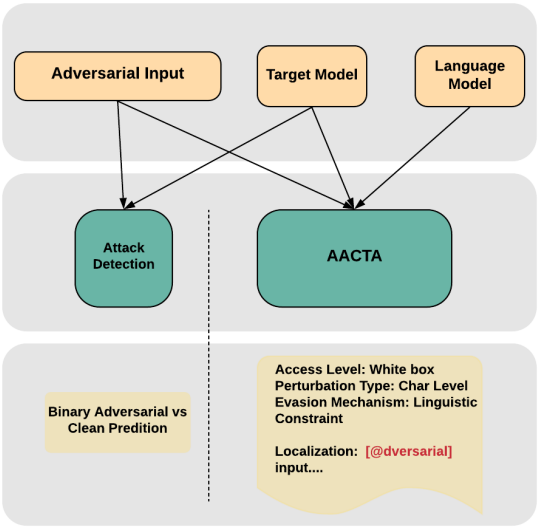

Figure 1: Illustration of AACTA compared to attack detection.

To address the potential threat of adversarial attacks in text, various defense methods have been proposed, such as training classifiers to determine whether a piece of text is perturbed by potential attackers (Zhou et al., 2019) and hiding the gradient of the model via prediction poisoning (Wallace et al., 2020; Orekondy et al., 2019). However, it is easier to defend against a threat when we know more about the attacker. For example, commercial model owners may need to take actions if the attackers have access to protected model weights. Similarly, model users may want to receive alerts when the system is attacked by elaborate attackers that employ language models with GPU resources. As an attempt to address this issue, previous work has shown that fine-grained analysis of adversarial images can reveal information about the attackers, such as the target labels of the attacks on image clas-

Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP, pages 69–78

sifiers (Pang et al., 2020a). However, this method requires access to all queries from the attacker and a known target model.

In this work, we take the next step towards understanding attackers through a task we call ***A**ttacker **A**ttribute **C**lassification via **T**extual **A**nalysis (AACTA)*. AACTA includes a number of subtasks, including (1) attacker access level estimation, to determine what level of access the attacker has to the model being attacked; (2) perturbation type analysis, to determine the type of perturbation imposed by the attackers; (3) evasion mechanism determination, to determine whether the attackers used any constraints on generating imperceptible perturbations; and (4) attack localization, to determine which words or characters were changed.

To achieve these identification tasks, we exploit the fact that adversarial examples must make changes to the input text and thus the feature space of the target model. We draw signals from these traces as inputs to our classifiers, and formulate the task above as a series of classification subtasks. We conduct classification and localization over adversarial examples generated against multiple target text classifiers from various datasets. For classification sub-tasks, our method yields 74% accuracy on determining whether the *access level* of the attack is black-box or white-box, 75% accuracy on determining whether the attacking algorithm's *perturbation type* is word or character-level, and 78% accuracy on inferring whether the attackers attempt to construct *evasion method* by imposing linguistic constraints during attack construction. For attack localization, our model achieved 0.74 class-averaged f1 on flagging perturbed tokens[1].

2 Attacker Attribute Classification

2.1 Problem Statement

Consider an input space X and an NLP model parameterized by θ that maps the input space to an output space, i.e. $f_\theta(X) \rightarrow Y$. We refer to models under adversarial attacks as *target models*. The attacker's goal is to modify the output of the target model by making minor modifications to the input. For a given input $x \in X$, we define adversarial perturbations as samples x_{adv} that are close to x but with a different output from the target model:

$$\{\Delta(x, x_{\text{adv}}) \leq r, f_\theta(x) \neq f_\theta(\text{x}_{\text{adv}})\}$$

[1]Accuracy and f1 scores reported above are averaged scores across datasets.

Note that there can be various definitions of "closeness", corresponding to different distance metrics Δ. An adversarial attack $g : X \rightarrow X$ maps each input to an adversarial perturbation or to itself if no valid perturbation is found. Different attacking algorithms may use different constraints for "closeness" and different searching methods for generating perturbations.

In the context of AACTA, we aim to step beyond flagging adversarial samples from unperturbed texts. Specifically, we formulate AACTA as a series of classification and localization subtasks. For classification tasks, our goal is to infer whether the adversarial algorithm used to construct a sample x_{adv} belongs to a set of functions $A_{property}$, where *property* is a human-interpretable characteristic that is informative about the attackers. We accomplish this objective by training a classifier $h(X_{\text{adv}}) \rightarrow y_{property}$. For attack localization, the goal is to determine how the adversarial instance x_{adv} relates to the unseen source example x, namely, which token(s) in the sequence are modified by the attackers.

2.2 Classification Tasks

To perform sequence-level classification, we train logistic regression classifiers over two types of representations: a standard BERT sentence representation and a new target model representation, consisting of the layer-wise mean and variance of the node activations in each layer of the model. We show that the latter representation performs significantly better than the BERT representation in classifying the following properties:

1. *Target Model Access Level*: In this setup, the attacks are partitioned by the level of access to the target model (e.g. model weights and prediction probabilities). We formulate this sub-task as black-box vs. white-box classification. We consider an attack black-box if the attacker does not have any information about the target model; otherwise we consider the attack white-box. In some setups, adversarial attacks that require predicted probabilities of target models are considered grey-box attacks. Here we treat these attacks as white-box to ensure a wide enough range of attack algorithms populate both classes.

2. *Perturbation Type*: In this setup, we partition the attacks by perturbation type. This could include the level of perturbations (character or

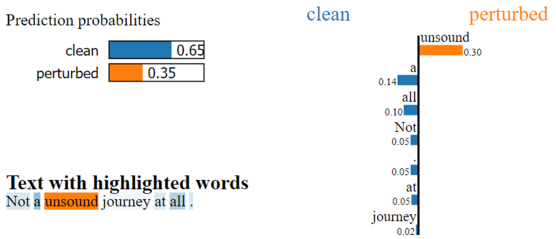

Figure 2: LIME explanation for a sentence-level classifier. The importance scores for the tokens could be used to localize perturbed tokens (see Algorithm 1). The localization algorithm does not fully depend on the classifier's ability to make correct predictions.

1: Input: Binary *sequence* classifier f, Interpretation framework LIME, Decision threshold $\epsilon \in [0, 1]$, Perturbed sequence x_{adv}
2: TokenScore $\leftarrow$ GetImportance(f,x_{adv}, LIME)
3: Sort tokens by their importance score
4: i $\leftarrow$ 0
5: **for** ;, i $<$ NumTokens, i++ **do**
6: **if** ImportanceOf(i) $\leq$ ImportanceOf(i-1)$\times\epsilon$ **then**
7: break
8: **end if**
9: **end for**
10: PerturbedTokens = SortedTokens[:i]
11: return PerturbedTokens

Algorithm 1: Algorithm for Locating Perturbed Token with Binary Adversarial-or-Not Classifier

word level), or more fine-grained classes of adversarial edits such as in-word shuffling, intrude, and character deletion (Eger and Benz, 2020). In our experiments, we separate attacks into two groups: word-level and character-level perturbations. The attacks are recognized as character-level attacks if the attacks impose perturbations on both character and token levels.

3. *Evasion Mechanism*: In this setup, the attacks are partitioned by the constraints used by attackers in order to generate more indistinguishable attacks. Such constraints could be making visually imperceptible (Eger et al., 2019) and grammatically coherent (Zhao et al., 2018) adversarial attacks, or simply constraining the number of perturbations allowed per sentence.

   In our experiment, we label the adversarial instances by whether the attackers impose linguistic constraints on ensuring the coherency of perturbed sentences. For example, such constraints could involve language models on word or sentence level, distance metrics in word embedding space, and other constraints such as sememe. Upon training and evaluating our detection classifiers, adversarial attacks that impose any of the linguistic constraints are treated as one class, and attacks that do not consider coherency of perturbed sentences are treated as the other class.

2.3 Attack Localization

For attack localization, we incorporate signals from a target model and an external language model with two different approaches.

For classifiers using an external language model signal, we first obtain contextualized word embeddings for each token in a sequence, and then train a binary perturbed-vs-clean *token-level* logistic regression classifier to determine if a token is perturbed or not. As BERT contextualized representation works on sub-word level, the detector model using BERT representation as input makes predictions on sub-word level as well. To evaluate BERT-based localizers in a more realistic setup, we applied a hierarchical tokenization scheme at inference time. Specifically, for each space-separated token in a potentially perturbed sequence, we flag the token as malicious if our binary classifier detects any of the subwords contained in the token to be perturbed.

Since the target model activation signal is obtained with respect to the whole sequence, there is no way to directly train an activation-based classifier that makes token-level predictions. Thus, we cast attack localization as interpreting a sequence level classifier, i.e. identifying tokens that are important to the classifier's decision that some sentence is adversarial. Specifically, we train a binary *sequence-level* clean-vs-perturbed classifier that aims at flagging perturbed sentences, then we apply LIME (Ribeiro et al., 2016) to obtain an importance score for each token with respect to the adversarial class. The exact heuristic used to identify perturbed token(s) with LIME is described in Algorithm 1. When obtaining token importance via LIME, we always consider important tokens that skew the detection model's prediction to the *adversarial* class. Thus, the localization algorithm could identify perturbed tokens even when the detection model makes an inaccurate prediction on the sequence level. Meanwhile, such an interpretation approach could be applied to any general binary-sequence-level classifier.

	Acc. Lvl	Evasion Mec.	Perturb. Type
BAE	bb	yes	word
DeepWordBug	bb	no	char
FasterGenetic	bb	yes	word
Genetic	bb	yes	word
Hotflip	wb	yes	char
IgaWang	bb	yes	word
Pruthi	bb	no	word
Pso	bb	yes	word
Textbugger	wb	yes	word
Textfooler	bb	yes	word
Viper	bb	no	char

Table 1: **Adversarial attacks included.** From left to right: access level (black-box vs. white-box, denoted by bb/wb in the table), evasion mechanism (with/without linguistic constrains), and perturbation type (word vs. character-level).

3 Experiment Setup

Datasets We generate adversarial text instances against the target models with open-source adversarial evaluation codebases, OpenAttack (Zeng et al., 2020) and TextAttack (Morris et al., 2020). The incorporated attacks are DeepWordBug (Gao et al., 2018), FasterGenetic (Jia et al., 2019), Genetic (Alzantot et al., 2018), HotFlip (Ebrahimi et al., 2018), IGA (Wang et al., 2019), Pruthi (Pruthi et al., 2019), PSO (Zang et al., 2020), TextBugger (Li et al., 2018), TextFooler (Jin et al., 2019), and Viper (Eger et al., 2019). Table 1 details the attack partitioning.

Target Models The adversarial samples are constructed against BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and XLNet models (Yang et al., 2019) over two tasks: abuse detection and sentiment classification. For abuse detection, the models are trained on Hatebase, Civil Comments[2], and Wikipedia (Wulczyn et al., 2017; Dixon et al., 2018) datasets; for sentiment analysis, the models are trained on SST (Socher et al., 2013) and Twitter Climate Change Sentiment[3] datasets.

Evaluation Metrics We evaluate our localizers on f1 scores, and all classification models (including classification models trained for BERT localizers) by class-balanced accuracy scores. For localizers specifically, all predictions are truncated to space-separated token level (as described in Section 2.3).

Hyper-parameters For the LIME-based localizer, we set the decision threshold ϵ to 0.1 for all experiments (see Algorithm 1). For all classification tasks, we train logistic regression models with standard scaled data, and tune the model's hyper-parameters using grid search with 3-fold cross-validation.

4 Localization Results

We cover empirical and qualitative results of our localization models in this section, and provide case studies of localized adversarial instances by both BERT and LIME-based localizers.

4.1 Empirical Performance

In our localization experiments, both types of localizers using BERT contextualized embeddings and target model activations outperform the random baseline significantly. Namely, our BERT-based and LIME-based localizers achieve 74% and 63% class-balanced f1-score on localizing perturbed tokens, while the random baseline is 53%. This shows that information of perturbed tokens could be retrieved from either external language models and target models. Meanwhile, although LIME-based localizers have a lower general f1 score than BERT-based localizers, LIME localizers achieve better accuracy at identifying the perturbed tokens. This is because the optimization objective of sequence-level classifiers only requires the model to identify at least one of the perturbed tokens in a sequence. Consequently, the classifier tends to give accurate but conservative decisions when flagging perturbed tokens. On the other hand, BERT-based localizer is directly trained on binary classification and thus has better-balanced performance.

For LIME-based localizers specifically, the performance trade-off between clean and adversarial classes of the inference process could be adjusted by tuning the threshold hyper-parameter as described in the previous section. For our experiment, all localizers were set to conduct inference with a threshold of 0.1.

4.2 Qualitative Analysis

BERT-based Localizer In practice, BERT-based localizers tend to flag regions of perturbed text instead of individual tokens, potentially due to the usage of contextualized embeddings as input.

[2] https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification

[3] https://www.kaggle.com/edqian/twitter-climate-change-sentiment-dataset

	Original Texts	Perturbed Texts	Bert Localizer Outputs	Actv Localizer Outputs
1	Vile and tacky are the two best adjectives to describe Ghost Ship .	Hateful and dorky are the two best adjectives to describe ghost ship.	Hateful and dorky are the two best adjectives to describe ghost ship.	Hateful and dorky are the two best adjectives to describe ghost ship.
2	Its gorg out! RT @@anggxo: This bitch thinks its june or something @BriiiXO http://t.co/mFVCmym	Its gorg outside! Ti @@ anggxo : this blackguard thinks its june or something @ briiixo http :// t. Co / mfvcmym	Its gorg outside! Ti @@ anggxo : this blackguard thinks its june or something @ briiixo http :// t. Co / mfvcmym	Its gorg outside! Ti @@ anggxo : this blackguard thinks its june or something @ briiixo http :// t. Co / mfvcmym
3	It 's difficult to say whether The Tuxedo is more boring or embarrassing – I 'm prepared to call it a draw .	Iẗ 'ŝ diffi̇culẗ to sây whetheř Ťhê Ťŭxeq̇ô îs moře borîng or embârrâsŝîñg – I 'ḿ prêp'ařêq̇ tô calḷ iẗ a dřâw .	Iẗ 'ŝ diffi̇culẗ to sây whetheř Ťhê Ťŭxeq̇ô îs moře borîng or embârrâsŝîñg – I 'ḿ prêp'ařêq̇ tô calḷ iẗ a dřâw .	Iẗ 'ŝ diffi̇culẗ to sây whetheř Ťhê Ťŭxeq̇ô îs moře borîng or embârrâsŝîñg – I 'ḿ prêp'ařêq̇ tô calḷ iẗ a dřâw .
4	Wake up bitches	Following up fairies	Following up fairies	Following up fairies

Table 2: Selected examples of localizer predictions. Dark red denotes perturbed or correctly flagged tokens, blue denotes false negative predictions, and green denotes false positive predictions.

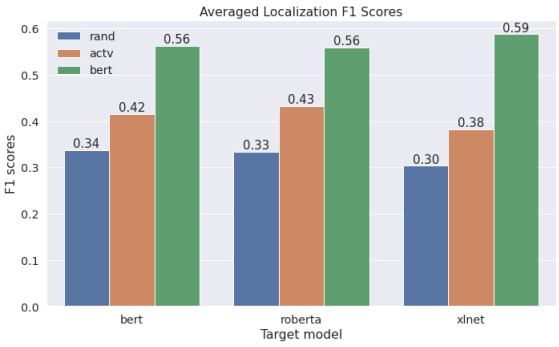

Figure 3: Average *adversarial* class f1 score over all datasets and target model combinations. Leftmost bars colored in blue denotes performance of a random token-level classifier.

For example (Table 2: instance 1), BERT localizer identified *"Hateful and dorky"* as consecutive perturbed tokens, although the word *"and"* is not perturbed. Similarly, in the perturbed phrase "*Iẗ 'ŝ diffi̇culẗ* to *sây*" in instance 3, BERT localizer flagged the word *"to"* as perturbed.

On the other hand, the auto-correction scheme utilized by the off-the-shelf BERT tokenizer sometimes hinders the performance of the localizer. For example, BERT tokenizer will convert the visually attacked token *"îs"* to the regular token *"is"*; this makes it challenging for the BERT-embedding-based classifier to pick up such perturbed tokens.

In general, BERT-based localizers tend to give more aggressive predictions regarding perturbed tokens, thus yielding more false positives, as demonstrated in instance 1. However, the localizer struggles for instances where the original text contains many low-frequency tokens. For example, in instance 2, the only correctly identified perturbed token is a character level perturbation on the word "*blackguard*." Perturbations on user ids and URLs are all overlooked.

LIME-based Localizer In contrast to the BERT-based localizer, the activation-based localizer achieves higher accuracy detecting perturbed tokens, as demonstrated empirically in Section 4.1. For example, in instance 1 of Table 2, the activation-based localizer did not yield false-positive labels on the phrase "*Hateful and dorky*." However, the more conservative decision made by the LIME-based localizer makes it more likely to miss perturbed tokens. In instance 4, the localizer failed to recognize "*fairies*" as a perturbed token.

For both localizers, conducting classification on noisy inputs becomes a challenging task. Instance 2 in Table 2 shows an example where the input text contains user ids and hyperlinks. In this case, both localizers struggle to identify the perturbed tokens, while BERT-based localizers have the least accurate predictions. This may be explained by the failure of the pre-trained language model to capture the text properties corresponding to specific datasets, while target models are adapted to each of the datasets during training. We provide further discussion on this phenomena in Section 6.2.

5 Attack Attribute Classification

We evaluate the performance of our classification task by class-balanced accuracy scores. The results for classification tasks are shown in Figure 4 and Figure 5. For almost all target models and datasets, classifiers using target model activation as input signals consistently outperform classifiers using

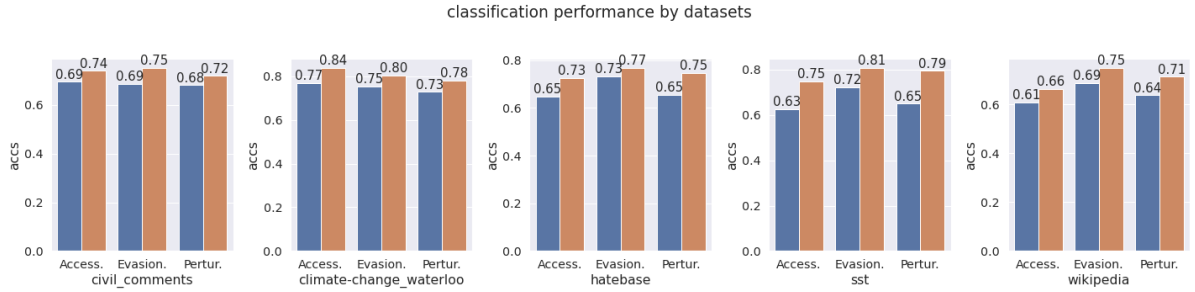

Figure 4: Accuracy score of each of the three classification task over all datasets. Blue bars denotes performance of BERT representation, orange bars denotes performance with target model activation. The three property being detected are access level, evasion mechanism and perturbation type.

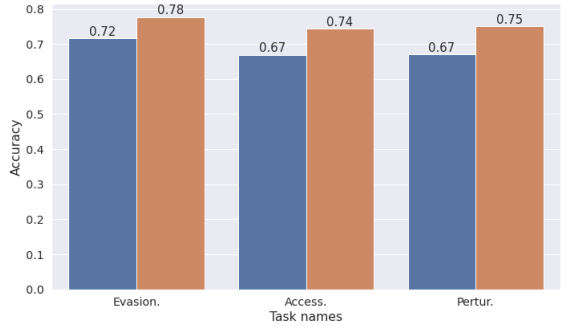

Figure 5: Average accuracy for different tasks across all datasets and target model combinations. The blue and orange bars denote BERT and activation-representation performance, respectively.

	wikipedia	sst	hatebase	climate.	civil.
civil.	57.4	59.0	54.1	58.5	na
climate.	57.2	57.6	55.3	na	60.1
hatebase	59.7	57.7	na	58.1	56.7
sst	58.4	na	55.1	62.5	60.6
wikipedia	na	58.2	61.0	59.8	62.8

Table 3: Averaged out-of-domain (dataset) generalization detection accuracy scores across all models for BERT-based classifiers.

BERT sentence representations.

Target Model Access Level On determining whether an adversarial attack is constructed with access to the target model, target model internal activation representations outperform BERT representations the most among the three tasks, demonstrating white and black box attacks will form different distributions in the representation space of target models. Interestingly, target model representation as a proxy to separate black and white-box attacks consistently achieve better performance in sentiment classification tasks (datasets) than abuse detection datasets.

Perturbation Type In classifying whether the attack algorithm operates on word or character-level, BERT and target model representation has the least significant performance gap among the three tasks. This shows pre-trained language models and target models capture information on the token level of perturbed texts to a similar extent. However, activation-based localizer still outperforms BERT-based localizer in all experiment setups, which could be explained by the phenomenon where general pre-trained language models fail to capture the nuances of specific datasets, as described in the previous section.

Evasion Mechanism We observe the highest accuracy for the task of determining whether attackers impose any linguistic constraints when curating adversarial samples. Similar to detecting black and white box attacks, detection models on sentiment classification datasets achieve higher scores in general than abuse detection datasets.

6 Discussion

This section is organized as follows: we first discuss out-of-domain performance of classifiers and localizers, then provide an analysis on the performance of BERT representations.

6.1 Out-of-Domain Generalization

We test our model's ability to generalize across target models and datasets by evaluating trained classifiers on (1) adversarial text instances targeted at different target models trained on the same dataset, (2) adversarial text instances targeted at the same target model trained on different datasets.

Generalization of BERT-based Classifiers Our results show BERT-representation-based classi-

	wikipedia	sst	hatebase	climate.	civil.
civil.	52.6	51.8	51.2	53.2	na
climate.	51.7	53.0	52.7	na	54.4
hatebase	51.4	50.9	na	52.4	51.5
sst	51.0	na	53.1	52.7	50.4
wikipedia	na	52.0	54.0	55.4	53.0

Table 4: Averaged out-of-domain (dataset) generalization accuracy scores across all models for activation-based classifiers

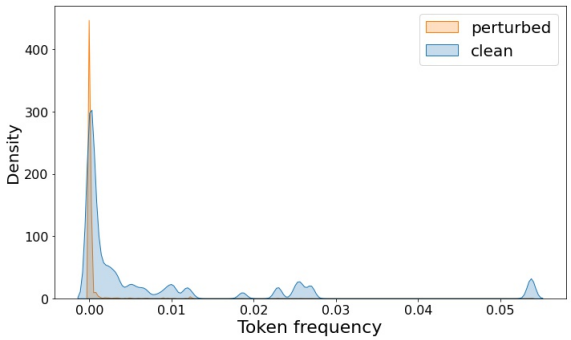

Figure 6: Frequency distribution of adversarial and clean tokens with respect to external corpus for all adversarial instances.

fiers demonstrate abilities to generalize to adversaries targeting different target models or datasets. Meanwhile, there is no significant difference between BERT-based classifiers' generalization performance to datasets for the same task and different tasks (i.e., abuse detection and sentiment classification).

Generalization of Activation-based Classifiers Compared to BERT-representation-based classifiers, classifiers that operate on target-model activation-representations do not generalize well to other datasets. In almost all cases, performance metrics for the classifier dropped to near-random performance on out-of-distribution data. This shows that although activation representations provide richer information about the attackers for in-distribution data, they are less dataset-agnostic than language-model representations.

Generalization of Localizers Similar to classifiers using two types of signals, BERT-based localizers outperform LIME (activation)-based localizers when given out-of-distribution data. When averaging over all out-of-domain datasets, the class-weighted f1 score of BERT-based localizer drops to 0.69, while performance of the LIME-based localizer drops to 0.28. This is consistent with the observations of classifiers using the two types of representations in previous sections: BERT representations tend to have better generalization ability across datasets, while signals from target models are highly dependent on specific domains.

6.2 BERT Feature Performance and Adversarial Token Frequencies

In sequence classification experiments, our proposed feature from target model activation statistics outperforms BERT representations. Similarly, previous works show that BERT does not work well in discriminating character-level attacks (Zhou et al., 2019). In this section, we demonstrate this might be due to perturbed tokens that are usually rare in natural texts; thus, pre-trained language models do not have enough signal to fully capture the feature of perturbed tokens.

To test the plausibility of this hypothesis, we apply an off-the-shelf word-frequency lookup library (Speer et al., 2018) to check the frequency of each token with respect to several external corpora[4], and show the distribution of token frequency with kernel density estimation. The plotted distribution of clean and adversarial tokens is shown in Figure 6. This gives some intuition on why target model activation achieves better results. Namely, tokens perturbed by attackers tend to have low occurrence in natural text, and thus forms out-of-distribution data to pre-trained language models. Thus, there might be more fine-grained signals that are lost during the encoding phase of BERT language model.

7 Related Work

This section is organized as follows: we first cover related literature in detecting and defending against adversarial input for deep neural networks. Then, we cover some closely connected lines of works and highlight their similarities and differences to our approach.

Adversarial Attacks for Deep Neural Networks Deep learning has achieved promising results in a wide range of domains such as CV and NLP. However, neural-network-based models are usually vulnerable to adversarial attacks. Although adversarial training could help, such a method is known to hurt the generalization ability and performance of the resulting models (Raghunathan et al., 2019) and

[4]Wikipedia, Google Books, Twitter and other commonly used corpora from various domains, see https://github.com/LuminosoInsight/wordfreq

requires additional data curation and training procedures. Exploiting the fact that adversarial examples are usually scattered in a different distribution than innocent inputs (Gong et al., 2017), people have achieved successful binary detection performance using external classifiers that directly operate on input data (Metzen et al., 2017; Feinman et al., 2017). Meanwhile, other works have shown that adversarial examples leave recognizable traces in latent spaces of the attacked models, enabling defensive methods that utilize model internals, such as using KNN on model states (Carrara et al., 2019).

Attack Detection in Text Different from detecting adversarial attacks in images, capturing text adversaries requires approaches that take into account the featurization of discrete text tokens. Previous works have shown that linear models that use BERT representation of each token as inputs could achieve good performance for word-level attacks and achieve competitive general detection accuracy over adversaries when combined with a spell checker (Zhou et al., 2019).

Obtaining Information about Attackers Via Adversarial Samples It is possible to learn even more about the attackers via analyzing a *batch* of adversarial exmaples, such as what class label the attacker interested in (Pang et al., 2020b). However, this method might not work for attacks that are only aimed at lowering the accuracy of target models. Additionally, our method could help in obtaining information about the attackers from more dimensions.

Deriving Interpretable Information From Latent Representations Previous studies have shown that interpretable linguistic information could be retrieved from representations of deep learning models, such as word embeddings and activation status of internal neurons (Belinkov, 2021). Our work borrows the general methodology from this literature but instead derives properties that are informative about the attackers from textural adversaries .

8 Conclusion

In this work, we introduce a new task, **A**ttacker **A**ttribute **C**lassification via **T**extual **A**nalysis (AACTA), that aims at deriving interpretable information about adversarial attack algorithms that are informative about the attackers. We demonstrate that characteristics of attacking algorithms leave traces in feature space of language models and target models, and that such information could be retrieved by training classifiers over latent representations.

Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA), agreement number HR00112090135. This work benefited from access to the University of Oregon high performance computer, Talapas.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Yonatan Belinkov. 2021. Probing classifiers: Promises, shortcomings, and alternatives. *arXiv preprint arXiv:2102.12452*.

Fabio Carrara, Fabrizio Falchi, Roberto Caldelli, Giuseppe Amato, and Rudy Becarelli. 2019. Adversarial image detection in deep neural networks. *Multimedia Tools and Applications*, 78(3):2815–2835.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Steffen Eger and Yannik Benz. 2020. From hero to zéroe: A benchmark of low-level adversarial attacks. *arXiv preprint arXiv:2010.05648*.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna

Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. *CoRR*, abs/1801.04354.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. *CoRR*, abs/2004.01970.

Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. 2017. Adversarial and clean data are not twins. *arXiv preprint arXiv:1704.04960*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4120–4133.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. TextBugger: Generating adversarial text against real-world applications. *CoRR*, abs/1812.05271.

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. *CoRR*, abs/1907.11692.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*.

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

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Prediction poisoning: Towards defenses against dnn model stealing attacks. *arXiv preprint arXiv:1906.10908*.

Ren Pang, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2020a. AdvMind: Inferring adversary intent of black-box attacks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20, page 1899–1907, New York, NY, USA. Association for Computing Machinery.

Ren Pang, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2020b. AdvMind: Inferring adversary intent of black-box attacks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1899–1907.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John C Duchi, and Percy Liang. 2019. Adversarial training can hurt generalization. *arXiv preprint arXiv:1906.06032*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. Luminosoinsight/wordfreq: v2. 2.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. *arXiv preprint arXiv:2004.15015*.

Xiaosen Wang, Hao Jin, and Kun He. 2019. Natural language adversarial attacks and defenses in word level. *CoRR*, abs/1909.06723.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399.

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.

Liping Yuan, Xiaoqing Zheng, Yi Zhou, Cho-Jui Hsieh, and Kai-Wei Chang. 2021. On the transferability of adversarial attacks against neural nlp models. In *EMNLP*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2020. OpenAttack: An open-source textual adversarial attack toolkit. *arXiv preprint arXiv:2009.09191*.

Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(3):1–41.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating Natural Adversarial Examples. In *International Conference on Learning Representations (ICLR)*.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4906–4915.