Sample Attackability in Natural Language Adversarial Attacks

Vyas Raina
University of Cambridge
vr313@cam.ac.uk

Mark Gales
University of Cambridge
mjfg@cam.ac.uk

Abstract

Adversarial attack research in natural language processing (NLP) has made significant progress in designing powerful attack methods and defence approaches. However, few efforts have sought to identify which source samples are the most attackable or robust, i.e. can we determine for an unseen target model, which samples are the most vulnerable to an adversarial attack. This work formally extends the definition of sample attackability/robustness for NLP attacks. Experiments on two popular NLP datasets, four state of the art models and four different NLP adversarial attack methods, demonstrate that sample uncertainty is insufficient for describing characteristics of attackable/robust samples and hence a deep learning based detector can perform much better at identifying the most attackable and robust samples for an unseen target model. Nevertheless, further analysis finds that there is little agreement in which samples are considered the most attackable/robust across different NLP attack methods, explaining a lack of portability of attackability detection methods across attack methods. [1]

1 Introduction

With the emergence of the Transformer architecture (Vaswani et al., 2017), natural language processing (NLP) models have demonstrated impressive performance in many tasks, ranging from standard sentiment classification (Abdullah and Ahmet, 2022) to summarisation (Boorugu and Ramesh, 2020) and translation (Yang et al., 2020). However, Goodfellow et al. (2014) demonstrated that deep learning models are susceptible to adversarial attacks, where carefully crafted small imperceptible changes applied to original, natural inputs can cause models to mis-classify. In response, extensive efforts have explored methods to combat the threat of adversarial attacks by training with adversarial examples (Qian et al., 2022) or building separate detection systems (Harder et al., 2021; Raina and Gales, 2022). However, little or no work has sought to determine which input samples are the most susceptible to adversarial attacks. Are certain input samples easier to adversarially attack and if so can we efficiently identify these *attackable* samples? The ability to identify the *attackable* and in converse the *robust* samples has applications in a range of sample-selection tasks. For example, in the field of active learning (Sun and Wang, 2010), the query system can be designed to select the most *attackable* samples. Similarly, knowledge of sample attackability is useful for weighted adversarial training (Kim et al., 2021), where the aim is to augment the training set with only the most *useful* adversarial examples.

In the image domain, Raina and Gales (2023) formally define the notion of sample attackability as the minimum perturbation size required to change a sample's output prediction from the target model. Running iterative adversarial attacks to determine this minimum perturbation size for a single sample is inefficient. Kim et al. (2021) use entropy (uncertainty) as a proxy function for sample attackability, but, Raina and Gales (2023) demonstrate that training a deep learning based classifier to predict the most attackable samples (and most robust samples) is the most effective method in the image domain. Therefore, this works extends the use of a deep learning based system to identify the most attackable and robust samples in NLP tasks. As a measure of a sample's attackability, it is challenging to define a sample's *perturbation size* for natural language. Following Raina and Gales (2023) in the image domain, this work uses the *imperceptibility* threshold in the definition of an adversarial attack as a measure of the perturbation size. To align with human perception, imperceptibility constraints for NLP aim to limit the seman-

[1] Code: https://github.com/rainavyas/nlp_attackability

Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023), pages 96–107

tic change in the text after an adversarial attack. These imperceptibility constraints can be grouped into two stages: 1) pre-transformation constraints (e.g. no stopword changes) that limit the set of acceptable adversarial examples; and 2) distance constraints that only allow for a subset of the acceptable adversarial examples, where the distance constraint explicitly restricts the *distance moved* by an adversarial example from the original example to satisfy a specified imperceptibility threshold. This distance can be measured for example using the Universal Sentence Encoder (Herel et al., 2022). A sample subject to a specific NLP attack method (with defined pre-transformation constraints) will have an associated set of acceptable adversarial examples. The attackability of the sample can thus be given by the smallest distance constraint imperceptibility threshold that at least one acceptable adversarial example in the set satisfies.

Default imperceptibility thresholds for the distance constraints proposed for NLP attack methods can often lead to unnatural adversarial examples (Morris et al., 2020). Hence, in this work, we use separate thresholds for defining *attackable* and *robust* samples. A sample's minimum perturbation size is required to be within a much stricter imperceptibility threshold to be termed *attackable*, whilst in converse a sample's minimum perturbation size has to be greater than a more generous imperceptibility threshold to be termed *robust*. The deep learning based attackability classifier proposed in Raina and Gales (2023) is successfully used to identify the attackable and robust samples for unseen data and unseen target models. However, in contrast to the image domain, it is found in NLP that the trained attackability detector fails to determine the attackable samples for different unseen NLP attack methods. This work extensively analyzes this observation and offers an explanation rooted in the inconsistency of imperceptibility definitions for different NLP attack methods.

2 Related Work

In the image domain Zeng et al. (2020) introduce the notion of sample attackability through the language of *vulnerability* of a sample to an adversarial attack. This vulnerability is abstractly defined as the distance of a sample to a model's decision boundary. Raina and Gales (2023) offer a more formal and extensive estimate of a sample's vulnerability/attackability by considering the smallest perturbation size, aligned with an adversarial attack's imperceptibility measure, to change a sample's class prediction. Other research in the field of *weighted adversarial training* (Kim et al., 2021), has also implicitly considered the notion of sample attackability. The aim in weighted adversarial training is train with the more *useful* adversarial examples, which are arguably sourced from the more *attackable* original samples. For example Kim et al. (2021) use model entropy to estimate this attackability, whilst Zeng et al. (2020) use model confidence and Raina and Gales (2023) are successful in using a deep-learning based estimator of attackability. In the field of NLP, little work has explored weighted adversarial training. Xu et al. (2022) propose a meta-learning algorithm to lean the importance of each adversarial example, but this has no direct relation to a source sample's attackability. Finally, in the field of active learning (Ren et al., 2020; Sun and Wang, 2010) there has also been implicit consideration of adversarial perturbation sizes as a measure of a sample's value. The aim in active learning is to select the most useful subset of samples in a dataset to train a model on. In the image domain, Ducoffe and Precioso (2018) propose the use of the smallest adversarial perturbation size for each sample to measure the distance to the decision boundary. However, there is no explicit consideration of sample attackability or design of an efficient method to identify the attackable samples.

3 Adversarial Attacks

In both the image and NLP domain, an untargeted adversarial attack is able to fool a classification system, $\mathcal{F}()$, by perturbing an input sample, $\mathbf{x}$ to generate an adversarial example $\tilde{\mathbf{x}}$ to cause a change in the output class,

$$\mathcal{F}(\mathbf{x}) \neq \mathcal{F}(\tilde{\mathbf{x}}). \quad (1)$$

It is necessary for adversarial attacks to be *imperceptible*, such that adversarial examples, $\tilde{\mathbf{x}}$ are not easily detectable/noticeable by humans. It is inefficient and expensive to rely on manual human measures of attack imperceptibility, so instead proxy measures are used to enforce imperceptibility of an adversarial attack. For images, the l_p norm is considered a good proxy for human perception of imperceptibility. However, in NLP it is more challenging to ensure imperceptibility. Despite earlier research introducing only visual constraints on the

adversarial attacks (Goyal et al., 2023; Gao et al., 2018; Ebrahimi et al., 2017; Pruthi et al., 2019; Tan et al., 2020; Li et al., 2018), e.g. number of words changed as per the Levenshtein distance, recent research considers more sophisticated measures seeking to measure the *semantic* change in text sequences (Li et al., 2020a; Jin et al., 2019; Ren et al., 2019; Wang et al., 2019; Garg and Ramakrishnan, 2020; Alzantot et al., 2018; Li et al., 2020b). In general, modern NLP imperceptibility constraints can be separated into two stages: *pre-transformation* constraints and *distance* constraints. Pre-transformation constraints typically limit the attack mechanism to encourage little change in semantic content. For example, stop-word transformations will be prevented or any word substitutions will be restricted to appropriate synonyms. A collection of pre-transformation constraints, as specified by a particular attack method, limit the available set, $\mathcal{A}$ of possible adversarial examples that can be considered for a specific sample, $\mathbf{x}$, such that

$$\tilde{\mathbf{x}} \in \mathcal{A}. \tag{2}$$

The *distance*-based constraints are further constraints that explicitly aim to limit the *distance* between the original sample $\mathbf{x}$ and the adversarial example, $\tilde{\mathbf{x}}$ to ensure a small perceived semantic change. This *distance* can be measured via a proxy function, $\mathcal{G}$,

$$\mathcal{G}(\mathbf{x}, \tilde{\mathbf{x}}) \leq \epsilon, \tag{3}$$

where ϵ represents the maximum imperceptibility threshold. A popular example of such a distance constraint is a limit on the cosine-distance in a sentence embedding space, e.g.,

$$\mathcal{G}(\mathbf{x}, \tilde{\mathbf{x}}) = 1 - \mathbf{h}^T \tilde{\mathbf{h}}, \tag{4}$$

where $\mathbf{h}$ and $\tilde{\mathbf{h}}$ are the normalized vector embedding representations of the word sequences $\mathbf{x}$ and $\tilde{\mathbf{x}}$.

4 Sample Attackability Definition

Sample attackability is concerned with how *easy* it is to adversarially attack a specific sample. The notion of sample attackability is formally introduced by Raina and Gales (2023), where a specific input sample, $\mathbf{x}_n$'s attackability for a specific model, $\mathcal{F}_k$ is given by the theoretical minimum perturbation size, $\hat{\delta}_n^{(k)}$ within which a sample can be successfully attacked. However, it is not simple to define the perturbation size for an adversarial attack in NLP. The simplest definition for the perturbation size, δ, for a specific attack method with a specific set of acceptable adversarial examples, $\mathcal{A}$ (Equation 2), is to use the distance-based proxy function, $\mathcal{G}$ (Equation 3), such that $\delta = \mathcal{G}(\mathbf{x}, \tilde{\mathbf{x}})$. Then the minimum perturbation size, $\hat{\delta}_n^{(k)}$ for sample n and model k is,

$$\hat{\delta}_n^{(k)} = \min_{\substack{\mathbf{x} \in \mathcal{A}, \\ \mathcal{F}_k(\mathbf{x}_n) \neq \mathcal{F}_k(\mathbf{x})}} \{\mathcal{G}(\mathbf{x}_n, \mathbf{x})\}. \tag{5}$$

We aim to use a sample's minimum perturbation size to classify it as *attackable*, *robust* or neither. Default distance-based imperceptibility constraints defined using $\mathcal{G}$ for various NLP attack methods can lead to unnatural adversarial examples and so we use separate and stricter thresholds for classifying samples as *attackable* or *robust*. Hence, as in Raina and Gales (2023), we define sample n as *attackable* for model k if the smallest adversarial perturbation is less than a strict threshold, $\mathbf{A}_{n,k} = (\hat{\delta}_n^{(k)} < \epsilon_a)$, where any sample that is not attackable can be denoted as $\bar{\mathbf{A}}_{n,k}$. Conversely, a sample is defined as *robust*, if its adversarial perturbation size is larger than a separate, but more generous (larger) set threshold, $\mathbf{R}_{n,k} = (\hat{\delta}_n^{(k)} > \epsilon_r)$.

It is informative to identify samples that are *universally* attackable/robust across different models. We can thus extend the definition for *universality* as follows. A sample, n, is **universally attackable** if,

$$\mathbf{A}_n^{(\mathcal{M})} = \bigcap_{k, \mathcal{F}_k \in \mathcal{M}} \mathbf{A}_{n,k}, \tag{6}$$

where $\mathcal{M}$ is the set of models in consideration. Similarly a sample is **universally robust** if, $\mathbf{R}_n^{(\mathcal{M})} = \bigcap_{k, \mathcal{F}_k \in \mathcal{M}} \mathbf{R}_{n,k}$. Note that all of the attackability definitions in this section are for a specific attack method (e.g. Textfooler), as definition of the perturbation size in Equation 5 uses the distance-based imperceptibility constraint, $\mathcal{G}$ specific to an attack method. Portability of these definitions and attackability detection models across attack methods is explored in Section 6.3.

5 Attackability Detector

The definition of attackable and robust samples uses the minimum perturbation size (as per a distance-based constraint) for an NLP adversarial attack on a sample. When trying to determine which samples are *attackable*, it is slow and expensive to run an adversarial attack iteratively to find

the minimum perturbation size. Further, often one may not have access to an unseen target model, $\mathcal{F}_t$ or even the target sample, n to perform an adversarial attack upon. Hence, in this setting, it is necessary to have a simple and efficient process that can determine whether samples in an unseen dataset are attackable for an unseen target model. Inspired by Raina and Gales (2023), this section describes a method to train a simple deep-learning attackability detector to identify the attackable and robust samples in an unseen dataset, for an unseen target model, $\mathcal{F}_t$. We give the deep-learning attackability detector access to a seen dataset, $\{\mathbf{x}_n, y_n\}_{n=1}^N$ and a set of seen models, $\mathcal{M} = \{\mathcal{F}_1, \ldots, \mathcal{F}_{|\mathcal{M}|}\}$, such that $\mathcal{F}_t \notin \mathcal{M}$. Each model can be represented as an encoder embedding stage, followed by a classification stage,

$$\mathcal{F}_k(\mathbf{x}_n) = \mathcal{F}_k^{(\texttt{cl})}(\mathbf{h}_{n,k}), \tag{7}$$

where $\mathbf{h}_{n,k}$ is the model encoder's embedding of $\mathbf{x}_n$. For each seen model in $\mathcal{M}$, a separate attackability detector can be trained. For a specific seen model, k, we can measure the attackability of each sample using the minimum perturbation size (Equation 5), $\{\hat{\delta}_n^{(k)}\}_{n=1}^N$. It is most efficient to exploit the encoder embedding representation of input text sequences, $\mathbf{h}_{n,k}$, already learnt by each model. Hence, each deep attackability detector, $\mathcal{D}_\theta^{(k)}$, with parameters θ, can be trained as a binary classification task to determine the probability of a sample being attackable for model k, using the encoder embedding at the input,

$$p(\mathbf{A}_{n,k}) = \mathcal{D}_\theta^{(k)}(\mathbf{h}_{n,k}). \tag{8}$$

Consistent with Raina and Gales (2023), we use a simple, single hidden-layer fully connected network architecture for each attackability detector, $\mathcal{D}$, such that,

$$\mathcal{D}_\theta(\mathbf{h}) = \sigma(\mathbf{W}_1\sigma(\mathbf{W}_0\mathbf{h})), \tag{9}$$

where $\mathbf{W}_0$ and $\mathbf{W_1}$ are the trainable parameters and $\sigma()$ is a standard sigmoid function. This collection of model-specific detectors can be used to estimate the probability of a new sample being attackable for an unseen target model, $\mathcal{F}_t$. It is most intuitive to take an expectation over the seen model-specific detector attackability probabilities,

$$p(\mathbf{A}_{n,t}) \approx \frac{1}{|\mathcal{M}|} \sum_{k,\mathcal{F}_k \in \mathcal{M}} p(\mathbf{A}_{n,k}). \tag{10}$$

Raina and Gales (2023) demonstrated that this estimate in the image domain does not capture the samples that are attackable specifically for the target model, $\mathcal{F}_t$'s specific realisation. Therefore, we seek instead to estimate the probability of a *universally attackable* sample (defined in Equation 6),

$$p(\mathbf{A}_n^{(\mathcal{M}+t)}) \approx \left[\frac{1}{|\mathcal{M}|} \sum_{k,\mathcal{F}_k \in \mathcal{M}} p(\mathbf{A}_{n,k})\right]^{\alpha(\mathcal{M})}, \tag{11}$$

where the parameter $\alpha(\mathcal{M})$ models the idea that the probability of sample being universally attackable should decrease with the number of models (note that this is empirically observed in Figure 1). An identical approach can be used to train detectors to give the probability of a sample being *universally robust*, $p(\mathbf{R}_n^{(\mathcal{M}+t)})$.

The attackability/robustness of samples can also be estimated using simple uncertainty based approaches, such as entropy (Kim et al., 2021) or a sample's class margin measured by model confidence (Zeng et al., 2020). These uncertainty measures can then also be compared to strict thresholds to classify samples as attackable or robust. Experiments in Section 6 compare the deep-learning based attackability detector to uncertainty-based attackability detectors. To assess which attackability detector performs the best in identifying attackable samples for the unseen target model, $\mathcal{F}_t \notin \mathcal{M}$, we consider four variations on defining a sample, n as attackable (Raina and Gales, 2023).

all- the sample is attackable for the unseen target model.

$$\mathbf{A}_{n,t} = (\hat{\delta}_n^{(t)} < \epsilon_a). \tag{12}$$

uni - the sample is universally attackable for the seen models and the unseen target model.

$$\mathbf{A}_n^{(\mathcal{M}+t)} = \mathbf{A}_{n,t} \cap \mathbf{A}_n^{(\mathcal{M})}. \tag{13}$$

spec - the sample is attackable for the target model but not universally attackable for the seen models.

$$\mathbf{A}_{n,t}^{\texttt{spec}} = \mathbf{A}_{n,t} \cap \bar{\mathbf{A}}_n^{(\mathcal{M})}. \tag{14}$$

vspec - a sample is specifically attackable for the unseen target model only.

$$\mathbf{A}_{n,t}^{\texttt{vspec}} = \mathbf{A}_{n,t} \cap \left(\bigcap_{k,\mathcal{F}_k \in \mathcal{M}} \bar{\mathbf{A}}_{n,k}\right). \tag{15}$$

Given that the deep learning based attackability detectors are trained to identify universally attackable samples (Equation 11), they are expected to perform best in the *uni* evaluation setting.

The corpus-level performance of an attackability detector for an unseen dataset can be reported using precision and recall. A selected threshold, β, is used to class the output of detectors, e.g. $p(\mathbf{A}_n^{(\mathcal{M}+t)}) > \beta$ classes sample n as attackable. The precision is $\texttt{prec} = \text{TP}/\text{TP+FP}$ and recall is $\texttt{rec} = \text{TP}/\text{TP+FN}$, where FP, TP and FN are standard counts for False-Positive, True-Positive and False-Negative. An overall score is given with the F1-score, $\text{F1} = 2 * (\texttt{prec} * \texttt{rec})/(\texttt{prec} + \texttt{rec})$. By sweeping the threshold β a full precision-recall curve can be generated and typically the threshold with the greatest F1-score is selected as an appropriate operating point.

6 Experiments

6.1 Setup

Experiments in this section aim to understand how well a deep-learning based detector, described in Section 5, performs in identifying attackable samples for an unseen dataset and an unseen target model, $\mathcal{F}_t$, where the detector only has access to a separate set of *seen* models, $\mathcal{M}$ during training. There are equivalent experiments looking to detect the most robust samples too. The performance of the deep learning based detector is compared to a baseline of uncertainty-based detectors (model confidence), inspired by Zeng et al. (2020), in which the samples with the most uncertain model predictions are identified as attackable and in converse the most certain samples are deemed to be robust. Specifically, two forms of uncertainty-based detectors are considered: 1) conf-u, where there is no access to the confidence from the unseen target model and so a sample's uncertainty is measured by an average of the confidence of the seen models, $\mathcal{M}$; and as a realistic reference we also consider 2) conf-s, where there is access to the target model output such that the target model's confidence is used directly as a measure of sample uncertainty.

Two popular natural language classification datasets are used in these experiments. First, the Stanford Sentiment Treebank2 dataset (sst) (Socher et al., 2013) is a movie review dataset with each review labelled as positive or negative. There are 6920 training samples, 872 validation samples and 1820 test samples. We also consider the Twitter Emotions dataset (Saravia et al., 2018), which categorizes tweets into one of six emotions: sadness, love, anger, surprise, joy and fear. This dataset contains 16,000 training samples, 2000 validation samples and 2000 test samples. For training of the attackability detectors, access was provided to only the validation data and hence the test data was used as an unseen set of samples to assess the performance of attackable sample detection.

These experiments work with four state of the art NLP transformer-based models: BERT (bert) (Devlin et al., 2018), XLNet (xlnet) (Yang et al., 2019), RoBERTa (roberta) (Liu et al., 2019) and Electra (electra) (Clark et al., 2020). Each model is of *base*-size (110M parameters). Finetuning on sst and twitter used ADAMW optimizer, 3 epochs and a learning rate of 1e-5. The performance of the models is given in Table 1. Three models (bert, xlnet, roberta) are treated as *seen* models, $\mathcal{M}$, that the attackability detector has access to during training. The electra model is maintained as the *unseen* target model, $\mathcal{F}_t \notin \mathcal{M}$ used only to assess the performance of the attackability detector.

Model	sst	twitter
bert	91.8	92.9
xlnet	93.6	92.3
roberta	94.7	93.4
electra	94.7	93.3

Table 1: Model Accuracy (%)

Four adversarial attack types are considered in these experiments: Textfooler (tf) (Jin et al., 2019), Bert-based Adversarial Examples (bae) (Garg and Ramakrishnan, 2020), Improved Genetic Algorithm (iga) (Wang et al., 2019) and Probability Weighted Word Saliency (pwws) (Ren et al., 2019). In the bae attack we consider specifically the BAE-R attack mode from the paper, where the aim is to **r**eplace tokens. For NLP adversarial attacks Section 3 discusses the nature of imperceptibility constraints, where constraints can either be *pre-transformation* constraints (Equation 2) or *distance*-based constraints (Equation 3). Table 2 summarises the constraints for each of the selected attack methods in this work. In the attackability detection experiments, the textfooler attack is treated as a *known* attack type, which the attackability detector has knowledge of during training, whilst the bae attack is an *unknown* attack type, reserved for evaluation of the detector to assess the portability

of the detector across attack methods. Evaluation of the attackability detector on the unseen datasets and the unseen target model (electra) with samples attacked by the known textfooler attack is referred to as *matched* evaluation, whilst samples attacked by the unknown bae attack is referred to as *unmatched* evaluation. The final two attack methods, pwws and iga, are used to further explore portability across attack methods in Section 6.3.

constraints	tf	bae	pwws	iga
no repeat tkn changes	✓	✓	✓	
no stopword changes	✓	✓	✓	✓
same part of speech swaps	✓	✓		
nearest neighbour syns swap	✓			✓
language model syns swap		✓		
wordnet syns swap			✓	
Universal Sentence Encoding	✓	✓		
Word Embedding Distance	✓			✓
% of words changed				✓

Table 2: Pre-transformation (top) and Distance-based (bottom) constraints for nlp adversarial attack methods.

6.2 Results

The first set of experiments consider the *matched* setting, where the known tf attack method is available at training time for the attackability detectors and also used to evaluate the attackability detectors. For each seen model, $\mathcal{M}$ (bert, xlnet, roberta), the tf attack method is used to determine the minimum perturbation size (as per distance-based constraints of the NLP attack method), $\hat{\delta}_n^{(k)}$, required to successfully attack each sample, n in the validation dataset (Equation 5). Note from Table 2 that this perturbation size is measured using the cosine distance for both word embeddings and Universal Sentence Encoder embeddings for the tf attack method. Using the sst data as an example, Figure 1 shows the fraction of samples, f that are successfully attacked for each model, as the adversarial attack constraint, ϵ_a is swept: $f = \frac{1}{N}\sum_n^N \mathbb{1}_{\mathbf{A}_{n,k}}$. Given this plot, we can sensibly define strict thresholds for attackability and robustness for the tf attack method: samples with a perturbation size below $\epsilon_a = 0.15$ are termed *attackable* and samples with a perturbation size above $\epsilon_r = 0.35$ are termed *robust*.

The aim now is to identify the attackable samples in the unseen test dataset that are vulnerable to attack as per the tf attack method for an unseen target model, $\mathcal{F}_t$ (electra). As described in Section 6.1, two baseline methods are considered: conf-u,

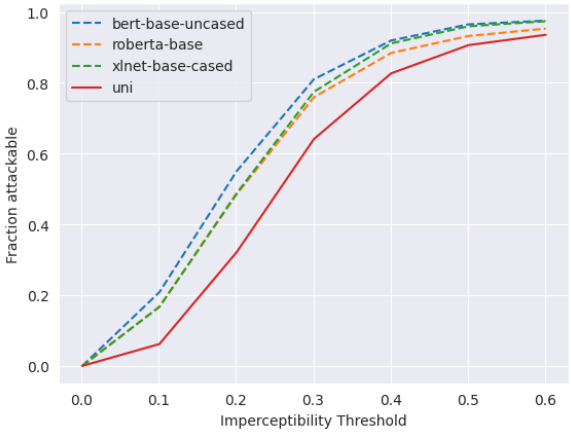


Figure 1: Fraction of *attackable* samples.

which has no knowledge of the target electra model and so uses the average confidence from the seen models, $\mathcal{M}$ (bert, xlnet and roberta); and conf-s, which has access to the predictions from the target model and so explicitly uses the target model's confidence to identify attackable samples. The method of interest in this work is the *deep*-learning based detector described in Section 5. Here, a single layer fully connected network (Equation 9) is trained with *seen* (bert, xlnet, roberta) model's final layer embeddings, using the validation samples in a binary classification setting to detect attackable samples. The number of hidden layer nodes for each model's FCN is set to the encoder embedding size of 768. Training of the FCNs used a batch-size of 8, 5 epochs, a learning rate of 1e-5 and an ADAMW optimizer. Table 3 shows the (best) F1 scores for detecting *attackable* samples on the unseen test data for the unseen target electra model, in the matched setting. Note that the scale of F1 scores can vary significantly between evaluation settings (*spec*, *vspec*, *uni* and *all*) as the prevalence of samples defined as *attackable* in a dataset are different for each setting and so it is not meaningful to compare across evaluation settings. Table 4 presents the equivalent results for detecting robust samples, where the definitions for each evaluation setting update to identifying *robust* samples ($\mathbf{R}_{n,k}$). For both the twitter and sst datasets, in detecting attackable samples, the deep detection method performs best in all evaluation settings, whilst for robust sample detection it performs significantly better in only the *uni* evaluation setting. Better performance in the *uni* setting is expected due to the deep detection method having been designed explicitly to detect universally at-

tackable samples (across models) (Equation 11), whilst for example the *conf-s* detection method has direct access to the target unseen model (electra) and so has the ability to perform competitively in the *spec* and *vspec* settings.

Setting		conf-s	conf-u	deep
all	sst	0.244	0.243	0.461
	twitter	0.457	0.457	0.516
uni	sst	0.103	0.110	0.281
	twitter	0.299	0.300	0.435
spec	sst	0.165	0.165	0.130
	twitter	0.220	0.222	0.236
vspec	sst	0.038	0.047	0.052
	twitter	0.062	0.063	0.055

Table 3: Attackable Sample Detection (F1) in matched setting.

Setting		conf-s	conf-u	deep
all	sst	0.448	0.449	0.476
	twitter	0.099	0.102	0.220
uni	sst	0.165	0.156	0.302
	twitter	0.025	0.028	0.091
spec	sst	0.340	0.340	0348
	twitter	0.088	0.082	0.206
vspec	sst	0.126	0.125	0.123
	twitter	0.025	0.015	0.053

Table 4: Robust Sample Detection in matched setting.

Figure 2(a-b) presents the full precision-recall curves (as described in Section 5) for detecting attackable samples in the *uni* evaluation setting, which the deep-learning based detector has been designed for. It is evident that for a large range of operating points, the deep detection method dominates and is thus truly a useful method for identifying attackable samples. Figure 2(c-d) presents the equivalent precision-recall curves for detecting robust samples. Here, although the deep-learning method still dominates over the uncertainty-based detectors, the differences are less significant. Overall, it can be argued that this deep learning-based attackability detector is better at capturing the features of the most attackable and robust samples in a dataset than standard uncertainty based methods.

Next we want to consider the *unmatched* setting, where the aim is to identify the attackable/robust samples in the test data, where the perturbation sizes for each sample are calculated using the *unknown* bae attack method. Referring to Table 2, the bae attack method has only one distance-based constraint (USE cosine distance) and so relative to the tf method with two distance based constraints, it is expected that with the definition of a sample's perturbation size, $\hat{\delta}_n^{(k)}$ (Equation 5), the bae attack method will have much smaller perturbation sizes than the tf perturbation size. This is demonstrated in Figure 4. Hence, for the bae attack to have a comparable number of attackable samples, the definition of the attackable threshold is adjusted to $\epsilon_a = 0.03$ and robustness threshold is kept at $\epsilon_r = 0.35$. Table 5 gives the F1 scores for detecting universal attackable/robust samples in the unmatched uni evaluation setting. In contrast to observations made in the image domain (Raina and Gales, 2023), here it appears that the deep detector fails to do any better than the uncertainty based detectors in identifying the attackable samples [2]. This suggests that the deep detector perhaps does not port over well to unknown attack methods (bae in this case) for NLP. The next section analyzes this observation further.

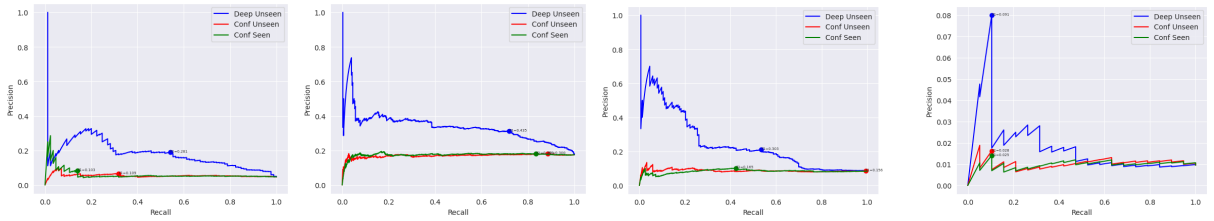
Uni setting		conf-s	conf-u	deep
Attackable	sst	0.555	0.555	0.555
	twitter	0.583	0.582	0.582
Robust	sst	0.02	0.129	0.250
	twitter	0.001	0.001	0.002

Table 5: Sample detection (unmatched setting).

6.3 Portability Analysis

In the above results it is shown that a deep-learning based method performs significantly better than uncertainty-based methods in identifying attackable/robust samples for an unseen target model with a known attack method (tf), but when used to identify samples for an unknown attack method (bae), it fails to port across (for attackable sample detection). This section aims to understand this observation in greater detail. First, for each model and dataset, the known tf attack and the unknown bae attacks were used to rank samples in the validation set by the minimum perturbation size, $\hat{\delta}_n$. In all cases the Spearman Rank correlation is lower than 0.2 for sst and twitter (Table 6). Hence it is not surprising that the results from the matched setting do not port easily to the unmatched setting.

[2] Interestingly, the deep detector does demonstrate some portability in identifying the most robust samples in the *uni* setting, suggesting that the robust samples are similar across different attack methods.



(a) sst-att (b) twitter-att (c) sst-robust (d) twitter-robust

Figure 2: P-R curves for detecting *universal* attackable/robust samples.

	bert	roberta	xlnet
sst	0.059	0.123	0.165
twitter	0.069	0.026	0.087

Table 6: Spearman rank correlation (tf, bae).

To attempt to understand the lack of agreement in sample perturbation sizes between the bae and tf attack methods, we consider two further nlp attack methods: iga and pwws. For each attack method, we use the default imperceptibility constraints (pre-transformation and distance-based constraints indicated in Table 2) and assess how effective these methods are in attacking the sst test set for each model. The results are presented in Table 7, where fooling rate is the fraction of correctly classified samples that are mis-classified after the adversarial attack. The final row considers the union of the different attack methods, where a successful attack by any one of the attack methods counts as a successful attack. It is surprising to note that although an individual attack method can achieve a fooling rate around 80%, the union of attack methods is nearer 100%. This demonstrates that different attack methods are able to attack a different set of samples, further highlighting that attackability/robustness of a sample is heavily dependent on the attack method.

	Fooling Rate (%)			
Attack	bert	xlnet	roberta	electra
tf, t	80.7	79.1	85.4	76.1
bae, b	63.9	60.8	65.3	60.7
pwws, p	78.2	70.8	74.9	73.3
iga, i	80.6	74.4	77.0	73.9
$t \cup b \cup p \cup i$	96.1	98.1	98.0	97.3

Table 7: Fooling rates with default constraints for attack methods

The interplay of sample attackability and the selected attack method can perhaps be explained by considering the imperceptibility constraints for each attack method. Equation 2 proposes the notion of an available set, $\mathcal{A}$ of possible adversarial examples that can exist for a specific source sample, $\mathbf{x}$, given the pre-transformation imperceptibility constraints. From Table 2 it is clear that the different attack methods have a different set of pre-transformation constraints, which suggests that each attack method can have non-overlapping available sets for a particular sample, $\mathbf{x}$, e.g. $\mathcal{A}^{\text{tf}} \neq \mathcal{A}^{\text{bae}}$. Hence, the smallest perturbation (as per the distance-based constraint) for a particular sample (Equation 5) can change significantly across attack methods, as there is simply a different set of available adversarial examples. Hence, it can be argued that an inconsistency in sample attackability across nlp adversarial attack methods is a consequence of the differences in the pre-transformation imperceptibility constraints.

7 Conclusions

Little research has sought to determine the level of vulnerability of individual samples to an adversarial attack in natural language processing (NLP) tasks. This work formally extends the definitions of sample attackability to the field of NLP. It is demonstrated that uncertainty-based approaches are insufficient in characterising the most attackable and the most robust samples in a dataset. Instead, a deep-learning based detector can be used to effectively to identify these attackable/robust samples for an unseen dataset and more powerfully for an unseen target model. However, it is also observed that different attack methods in natural language have a different set of imperceptibility constraints, leading to a lack of consistency in determining sample attackability across different attack methods. As a consequence, the success of a deep-learning based attackability detector is limited to the attack method it is trained with.

8 Limitations

This work introduced a powerful attackability detector but also demonstrated that its success is limited to a *matched* setting, where the same attack method is used in both training and evaluation of the detector. A second limitation with this work is that all experiments were carried out on natural language classification tasks. It would be useful in the future to extend these experiments to sequence-to-sequence tasks to have a more comprehensive set of results.

9 Ethics and Broader Impact

Adversarial attacks by nature can be of ethical concern, as malicious users can exploit theoretical adversarial attack literature to develop harmful tools to mis-use deployed deep learning systems. However this work does not aim to propose any new adversarial attack techniques, but instead considers a method to identify the most vulnerable/attackable samples. Hence, there is no perceived ethical concern related to this specific piece of work.

10 Acknowledgements

This paper reports on research supported by Cambridge University Press & Assessment (CUP&A), a department of The Chancellor, Masters, and Scholars of the University of Cambridge.

References

Tariq Abdullah and Ahmed Ahmet. 2022. Deep learning in sentiment analysis: Recent architectures. *ACM Comput. Surv.*, 55(8).

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. *CoRR*, abs/1804.07998.

Ravali Boorugu and G. Ramesh. 2020. A survey on nlp based text summarization for summarizing product reviews. In *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 352–356.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/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.

Melanie Ducoffe and Frédéric Precioso. 2018. Adversarial active learning for deep networks: a margin based approach. *CoRR*, abs/1802.09841.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for NLP. *CoRR*, abs/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. *CoRR*, abs/1801.04354.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples.

Shreya Goyal, Sumanth Doddapaneni, Mitesh M. Khapra, and Balaraman Ravindran. 2023. A survey of adversarial defences and robustness in nlp.

Paula Harder, Franz-Josef Pfreundt, Margret Keuper, and Janis Keuper. 2021. Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain. *CoRR*, abs/2103.03000.

David Herel, Hugo Cisneros, and Tomas Mikolov. 2022. Preserving semantics in textual adversarial attacks.

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.

Minseong Kim, Jihoon Tack, Jinwoo Shin, and Sung Ju Hwang. 2021. Entropy weighted adversarial training.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020a. Contextualized perturbation for textual adversarial attack. *CoRR*, abs/2009.07502.

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

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. BERT-ATTACK: adversarial attack against BERT using BERT. *CoRR*, abs/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. *CoRR*, abs/1907.11692.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. *CoRR*, abs/1905.11268.

Zhuang Qian, Kaizhu Huang, Qiu-Feng Wang, and Xu-Yao Zhang. 2022. A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies.

Vyas Raina and Mark Gales. 2022. Residue-based natural language adversarial attack detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Vyas Raina and Mark Gales. 2023. Identifying adversarially attackable and robust samples.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2020. A survey of deep active learning. *CoRR*, abs/2009.00236.

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, Florence, Italy. Association for Computational Linguistics.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

Li-Li Sun and Xi-Zhao Wang. 2010. A survey on active learning strategy. In *2010 International Conference on Machine Learning and Cybernetics*, volume 1, pages 161–166.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2920–2935, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

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

Jianhan Xu, Cenyuan Zhang, Xiaoqing Zheng, Linyang Li, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2022. Towards adversarially robust text classifiers by learning to reweight clean examples. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1694–1707, Dublin, Ireland. Association for Computational Linguistics.

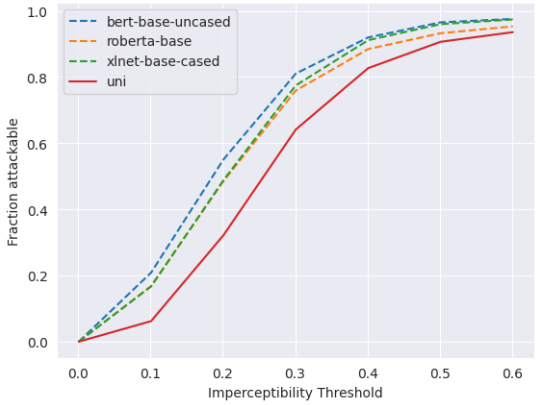
Shuoheng Yang, Yuxin Wang, and Xiaowen Chu. 2020. A survey of deep learning techniques for neural machine translation. *CoRR*, abs/2002.07526.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

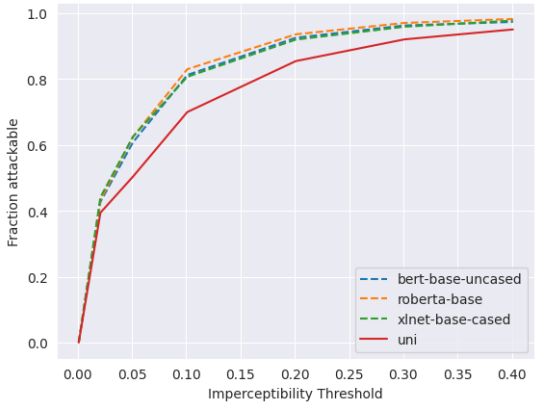
Huimin Zeng, Chen Zhu, Tom Goldstein, and Furong Huang. 2020. Are adversarial examples created equal? A learnable weighted minimax risk for robustness under non-uniform attacks. *CoRR*, abs/2010.12989.

Appendix

A Full set of empirical results

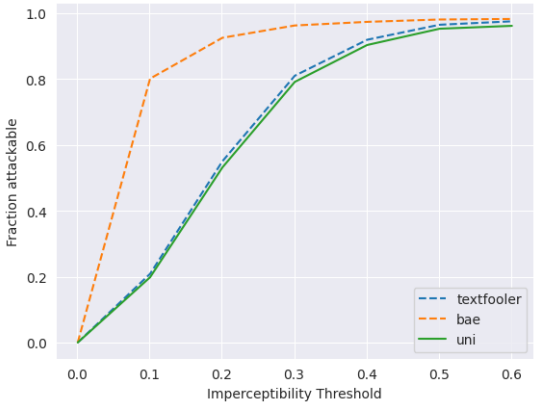


(a) textfooler

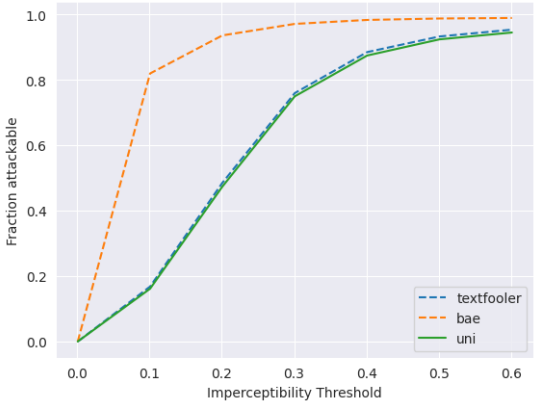


(b) bae

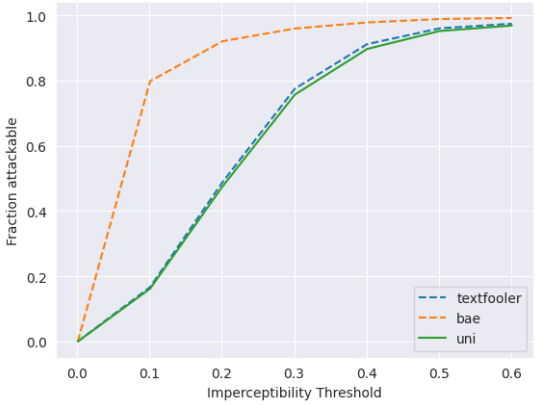
Figure 3: Fraction of samples classed as adversarially attackable across model architecture with increasing imperceptibility threshold as per distance-based constraint (sst).



(a) bert

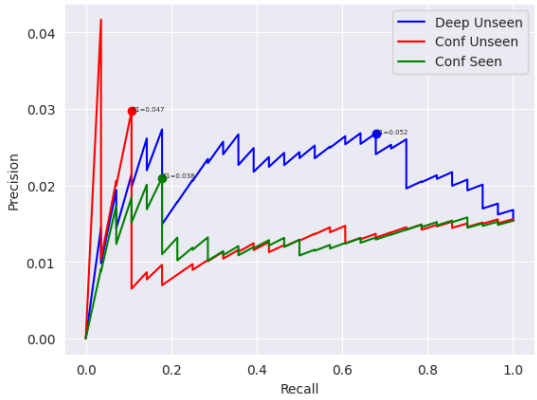


(b) roberta

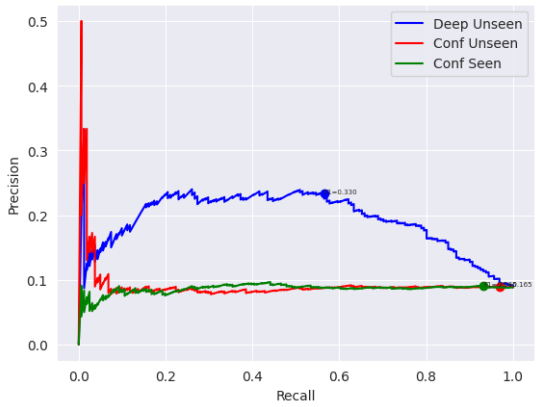


(c) xlnet

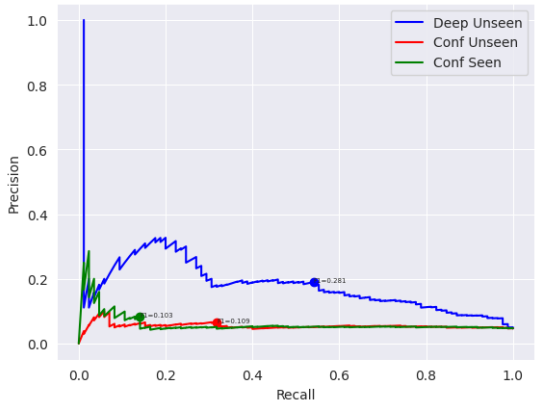
Figure 4: Fraction of samples classed as adversarially attackable across attack method with increasing imperceptibility threshold as per distance-based constraint (sst).



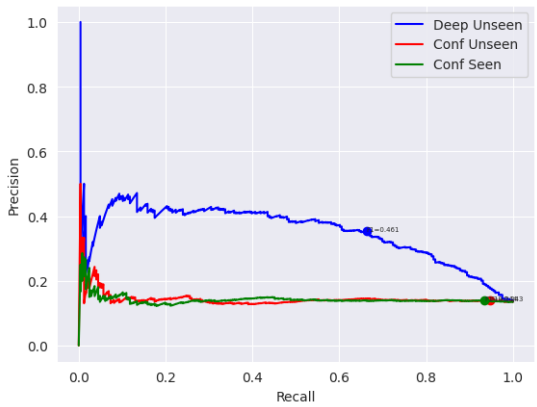
(a) Very Specific



(b) Specific

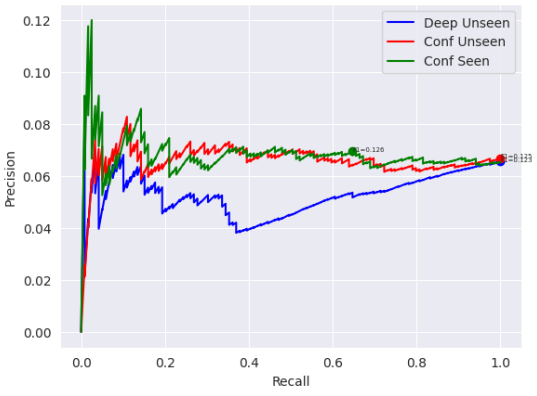


(c) Universal

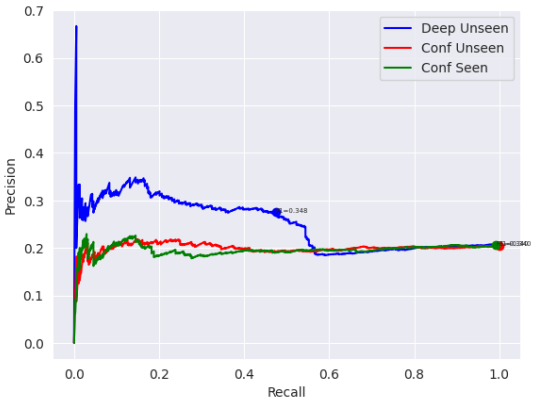


(d) All

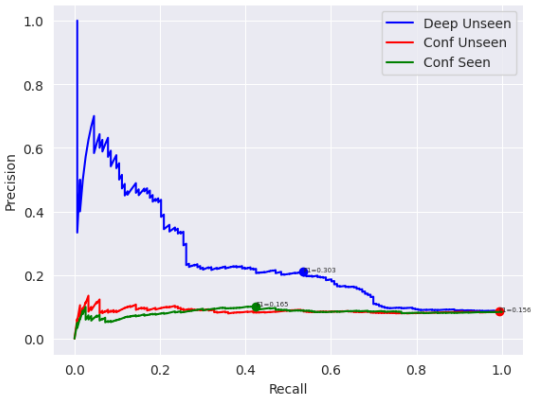
Figure 5: PR curves: Attackable Sample Detection (sst)



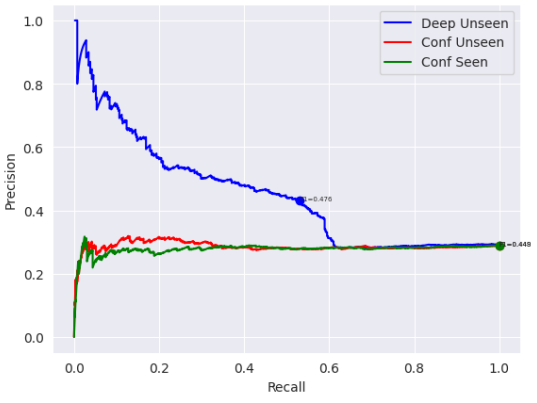
(a) Very Specific



(b) Specific



(c) Universal



(d) All

Figure 6: PR curves: Robust Sample Detection (sst)