# A Constraint–Enforcing Reward for Adversarial Attacks on Text Classifiers

**Anonymous submission**

## Abstract

Text classifiers are vulnerable to adversarial examples — correctly-classified examples that are deliberately transformed to be misclassified while satisfying acceptability constraints. The conventional approach to finding adversarial examples is to define and solve a combinatorial optimisation problem over a space of allowable transformations. While effective, this approach is slow and limited by the choice of transformations. An alternate approach is to directly generate adversarial examples by fine-tuning a pre-trained language model, as is commonly done for other text-to-text tasks. This approach promises to be much quicker and more expressive, but is relatively unexplored. For this reason, in this work we train an encoder-decoder paraphrase model to generate a diverse range of adversarial examples. For training, we adopt a simple policy gradient algorithm and propose a constraint-enforcing reward that promotes the generation of valid adversarial examples. Experimental results over two text classification datasets show that our model has achieved a higher success rate than the untrained paraphrase model, and overall has proved more effective than other competitive attacks. Finally, we show how key design choices impact the generated examples and discuss the strengths and weaknesses of the proposed approach.

## 1 Introduction

Adversarial attacks cause a *victim model* — an attacked machine learning model — to make a specific mistake. These attacks occur across domains, pose a real-world security threat[1] and are increasingly well-studied (Biggio and Roli, 2018; Zhang et al., 2020). In this paper we study adversarial attacks on text classifiers; where an adversary takes a correctly-classified *original* example and perturbs it to create an incorrectly-classified *adversarial* example. The adversarial example must typically meet some acceptability constraints (e.g., a maximum edit distance from the original, preserving semantic meaning, gramaticallity), although there is no general consensus on these (Morris et al., 2020b).

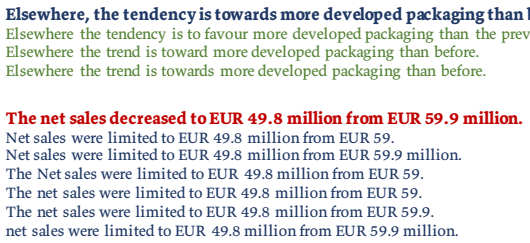



Figure 1: Examples of successful adversarial attacks against a sentiment classifier obtained with the proposed approach. On top, the adversarial examples flip the sentiment from the original neutral (blue) to positive (green), and on bottom, sentiment goes from the original negative (red) to neutral (blue).

How are text adversarial examples found? The predominant approach is to repeatedly modify tokens until the predicted label changes (Zhang et al., 2020). Attacks taking this approach, known as token-modification attacks (Roth et al., 2021), find adversarial examples by solving a constrained combinatorial optimisation problem. First, they define the success condition, the constraints and the allowed transformations, and then they use a search algorithm to seek a solution (Morris et al., 2020a). While effective, these attacks also have major downsides. Firstly, they are slow: the computational budget heavily impacts their success rate, with high-performing search algorithms requiring many victim model queries per example, particularly for long texts (Yoo et al., 2020). Secondly, their allowed token-level transformations limit their search space, largely preventing complex transformations like paraphrasing or style change.

[1] For example, (Wallace et al., 2020) attacked Google Translate with adversarial examples, causing vulgar outputs, word flips, and dropped sentences.

Alternatively, the adversarial example task can be formulated as a text-to-text problem, with original examples as input and adversarial examples as output. It could then be straightforwardly approached with seq2seq models, as done for other text-to-text tasks like summarisation or translation. This approach enjoys several principled advantages over token-modification attacks. Firstly, once trained, finding adversarial examples is much faster (in the order of a few milliseconds, rather than minutes or even hours). Additionally, through beam search or sampling, this approach can easily generate multiple adversarial examples per given input, while also controlling their diversity, tonality, or other characteristics. Finally, a seq2seq approach is also intrinsically more flexible as it is not limited by a rigid set of allowed transformations.

On the other hand, the main challenge of this approach is that it is notoriously difficult to train a model to generate controlled text. The training process can be unstable and time-consuming, and the generated text can be ungrammatical, irrelevant, nonsensical, unnatural, bland, repetitive, or incoherent (Holtzman et al., 2020; Hu et al., 2017; Wong, 2017). For our task there is an additional challenge: the generated text must change the victim model's predicted label while not violating any constraint.

For these reasons, in this paper we propose fine-tuning a pre-trained encoder-decoder paraphrase model so that it produces adversarial examples instead of paraphrases. We fine-tune using a reinforcement learning (RL) policy-gradient algorithm — REINFORCE with baseline (Williams, 1992) — and attack a sentiment classifier. For training, we propose an original reward function that both incentivises adversarial examples and penalises any violation of the constraints. To improve generated text coherence, our loss function includes a Kullback-Leibler (KL) divergence term (Kullback and Leibler, 1951) that limits parameter drift from the pre-trained paraphrase model. The attack requires the victim model's prediction confidence, but no other information, which makes our attack either a *grey-box* (Biggio and Roli, 2018) or a *black-box* attack (Zhang et al., 2020), depending on the definition.[2]

We have evaluated the proposed approach on two sentiment analysis datasets, reporting the attack success rates and the diversity of the generated adversarial examples across four different decoding methods and two training temperatures. The results show that the the proposed approach has been able to generate numerous and diverse adversarial examples, with success rates much higher than for the pre-trained paraphraser and comparable token-modification attacks. In summary, this paper makes the following key contributions:

1. an approach for the generation of adversarial attacks to text classifiers based on a pre-trained paraphraser and reinforcement learning;

2. a constraint-enforcing reward function that incentivises adversarial examples and penalises constraint violations;

3. experimental results on two text classification datasets showing the effectiveness of the proposed approach, and a comprehensive analysis and discussion.

## 2 Related Work

For ease of reference, we can divide the literature on text classification adversarial attacks into token-modification attacks and generative attacks.

**Token-modification attacks.** The vast majority of existing text adversarial attacks are token-modification attacks. They consist of four main components: a goal function, a set of allowed transformations, a set of constraints that must be satisfied, and a search method (Morris et al., 2020a). These approaches typically create adversarial perturbations by applying repeated token-wise transformations, such as character replacements (Ebrahimi et al., 2014) or synonym swaps (Ren et al., 2019)[3]. A detailed description of these attacks is not relevant to our work, so the reader can refer to a recent survey (Roth et al., 2021) for further details.

**Generative attacks.** Some previous work has attempted to train a variety of generative models to produce adversarial examples. For example, long short-term memory variants have been used by Iyyer et al. (2018) to paraphrase a sentence in

[2]These assumptions are not unrealistic: for example, most pre-trained models on the Hugging Face Model Hub report both predictions and confidences.

[3]These approaches typically produce one adversarial example per original. TextAttack (Morris et al., 2020b) — the most popular library for token-modification attacks — has been set up to only return at most one adversarial example per original, so this is what we have used throughout this paper. While more could be searched for, it has not been well studied how they could be found efficiently in incremental time.

the form of a parse template, and by Vijayaraghavan and Roy (2019) to perturb examples. A feed-forward network was used by Lu et al. (2022) to generate distracting answers in a multiple-choice visual question answering task. Other work has attempted to use GANs and autoencoders (Zhao et al., 2018; Ren et al., 2020; Wong, 2017). However, this line of approach has not been widely pursued, probably due to training difficulties. For example, Wong (2017) has reported widespread issues such as mode degeneracy, semantic divergence and reward hacking.

Since their introduction, transformers have become a ubiquitous encoder-decoder architecture in contemporary natural language processing. They are typically trained with transfer learning, first solving a large-scale pre-training task (typically unsupervised or self-supervised), and then fine-tuning on the target task. Large transformers such as T5 (Raffel et al., 2020) currently achieve state-of-the-art performance in many text-to-text tasks. Despite this success, no previous work we are aware of has attempted to fine-tune a pre-trained paraphrase model for adversarial example generation, as the proposed approach does. The closest works are Gan and Ng (2019), who create a dataset of adversarial paraphrases manually, and Qi et al. (2021), who use a pre-trained text style transfer model, but do not fine-tune it.

# 3 Proposed Approach

## 3.1 Overview

Our overall goal is to fine-tune a pre-trained paraphrase model with a reinforcement learning objective so that it can learn to generate adversarial examples. We use a T5 transformer as the base pre-trained model.

During each training epoch, we generate one paraphrase per original example and collate them into batches of training data. The batches are used to compute a loss function (Section 3.2), which incorporates both a reward function (Section 3.3) and a baseline (Section 3.4). We use a set of constraints (Section 3.5) to determine if the generated text is valid, and examples that fail receive zero reward. Figure 3 shows the overall setup.

During validation, we generate a set of adversarial example candidates for each original example, using one of four decoding methods (Section 4.2). We call these paraphrases the set of *adversarial example candidates*, and consider the attack successful if at least one meets the given constraints. The attack success rate is simply computed as the ratio between the number of successful attacks and the number of original examples. Generating more paraphrases can obviously improve the attack success rate, but the generation takes longer and the memory requirements increase. As an effective trade-off between these factors, we have chosen to generate $n = 48$ paraphrases per original example. The same procedure is used at test time. During validation, we also update the reward baseline with the average per-example reward across the candidate set. Training is stopped once the validation set performance improvement drops below a threshold, or after a maximum number of epochs (full details are available in Appendix A).

## 3.2 Loss function

Let us have an input data distribution $\mathcal{D}$, an original example $x \in \mathcal{D}$, and a pre-trained paraphrase model with parameters $\theta$. Given $x$, the model generates a paraphrase $x'$ with $T$ tokens with probability $p_\theta(x'_t|x'_1, \ldots, x'_{t-1}, x), t = 1 \ldots T$. We note this predictive distribution as $\rho$ for simplicity. Our agent attempts to learn a policy $\pi$, still parametric in $\theta$ and initially equal to $\rho$, that can create adversarial examples, for which we have a reward function $r$ that scores success and failure. Training aims to optimise $\pi$ to maximise the expected value of $r$:

$$\mathbb{E}_\pi(r) = \mathbb{E}_{x\sim\mathcal{D},x'\sim\pi(x)} r(x, x')$$

To optimise using gradient descent, the gradient $\Delta_\theta \mathbb{E}_\pi(r)$ is required, for which an estimator is provided by the policy gradient theorem (Sutton et al., 1999):

$$\Delta_\theta \mathbb{E}_\pi(r) = -r \sum_{t=1}^{T} \frac{\partial}{\partial\theta} \log \pi(x'_t|x'_1, \ldots, x'_{t-1}, x) \tag{1}$$

where $x'$ is a sampled sequence (using any of a number of sampling methods). The above is the REINFORCE estimator (Williams, 1992). Using an automatic differentiation framework we can convert this into a loss function:

$$\begin{aligned} L_{RF} &= -r \sum_{t=1}^{T} \log \pi(x'_t|x'_1, \ldots, x'_{t-1}, x) \\ &= -r \log \pi(x'|x) \end{aligned} \tag{2}$$

This estimator is unbiased, but it typically exhibits a large variance, which causes slow and unstable

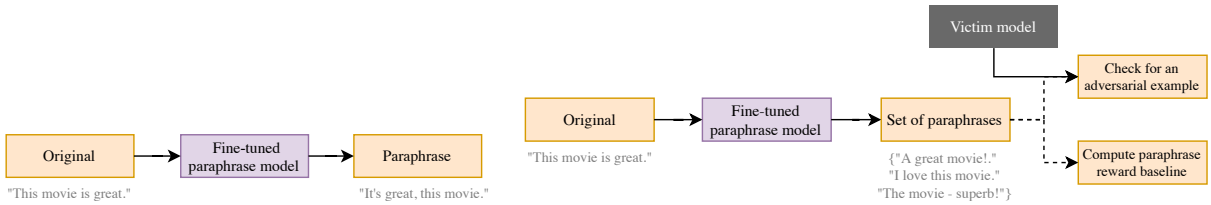



Figure 2: Sample generation during training and validation. (a) During training, we generate one paraphrase per original example, decoding with nucleus sampling. (b) During validation, we generate a set of paraphrases per original example, decoding with one of four methods (Section 4.2). We then check if any paraphrase in the set is a successful adversarial example, and also use the set (for the training split) to update the reward baseline (Section 3.4).

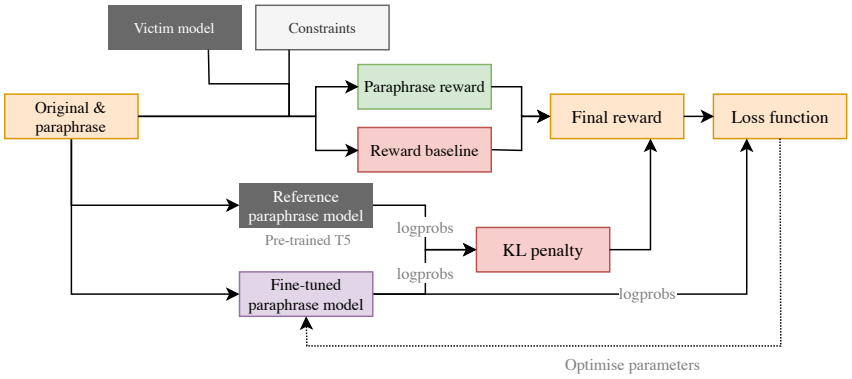



Figure 3: A diagram of the training approach. As input, training uses batches of *(original, paraphrase)* pairs. The parameters are updated using a REINFORCE with baseline algorithm. The overall loss function depends on the reward function, the baseline, the constraints, and the KL divergence penalty, which compares the probabilities computed by the fine-tuned and pre-trained paraphrase models.

learning. The variance can be reduced by subtracting a baseline, $b$, from $r$:

$$L_b = -(r - b) \log \pi(x'|x) \quad (3)$$

provided $b$ is highly correlated with $r$. This estimator is biased in the case that $b$ depends on $x'$ (Williams, 1992), but typically delivers improved training speed and stability.

We also would like to prevent the trained distribution, $\pi$, from diverging too much from the original predictive distribution, $\rho$, since that is likely to affect the coherence and paraphrase quality of the generated text. Following previous work (Jaques et al., 2017; Ziegler et al., 2019) we add a KL divergence penalty, $D_{KL}$, to discourage this behaviour. The modified reward function, after the baseline and the KL divergence term are incorporated, becomes:

$$R(x, x') = r(x, x') - b(x) - \beta D_{KL} \quad (4)$$

where $\beta$ is a scaling constant, and:

$$D_{KL} = \mathbb{E}_{x \sim \mathcal{D}, x' \sim \pi(x)}[\log \pi(x'|x) - \log \rho(x'|x)] \quad (5)$$

This leads to the overall loss function:

$$\mathcal{L} = -R(x, x') \log \pi(x'|x) \quad (6)$$

Finally, to prevent longer sequences from being unfairly penalised, we normalise the log probability terms $\log \pi(x'|x)$ and $\log \rho(x'|x)$ in (5) and (6) by dividing each by the generated sequence length, $T$.

### 3.3 Paraphrase reward

Let $f$ be the probability output by the victim classifier, $x$ be the original example with label $y$, and $x'$ be a paraphrase. Let $V(x, x') = f(x)_y - f(x')_y$ be the degradation in confidence in $y$ that $x'$ induces in $f$. Then the paraphrase reward to use in (4) is:

$$r(x, x') = \max(0, \min(\alpha, \eta\delta(x, x')V(x, x'))) \quad (7)$$

where $\alpha$ is an upper bound, $\eta$ a scalar multiplier, and $\delta(x, x')$ a Dirac delta function that is 1 if the constraints (Section 3.5) are met, and 0 otherwise.

### 3.4 Reward baseline

As shown in Equation 4, the gradient estimator requires a baseline $b$ for the reward. We use a per-example baseline, $b(x)$, and define it as the average reward of the set of adversarial example candidates generated for each $x$ in the training set. The baseline is updated in each validation phase (see Figure 2b). Intuitively, the $b(x)$ baseline is high when the model can easily generate adversarial examples for $x$, and low when it cannot.

### 3.5 Adversarial example constraints

In addition to switching the predicted label, an adversarial example should both preserve meaning (Michel et al., 2019) and be linguistically acceptable. We enforce these principles by using the following constraints:

**Retains the true label.** The original and paraphrase must have the same ground-truth label. Since the ground-truth label of the paraphrase is latent, this constraint is failed if the paraphrase contradicts the original with a probability $\geq 0.2$ according to a natural language inference pre-trained model.

**Is semantically consistent.** The original and paraphrase must have (broadly) the same semantic content. To assess this, we extract sentence embeddings of both using a pre-trained Siamese-BERT model (Reimers and Gurevych, 2019), compute their cosine similarity, and impose a minimum threshold of 0.8.

**Is linguistically acceptable.** Paraphrases should be acceptable sentences. This constraint is met only if the generated sentence is deemed linguistically acceptable with a probability $\geq 0.5$, according to a pre-trained language model.

Through trial and error, we decided to also introduce two additional constraints to prevent the generation of undesirable solutions:

**The sentence length is similar.** To prevent the generation of very short sentences, we require the original and paraphrase to have sentence length within 30 characters of each other.

**Avoids linking contrast phrases.** Regardless of the true class, the model can "soften" the generated paraphrase by starting or ending it with a linking contrast phrase, such as "however" or "nonetheless" (see Table 3). To encourage the generation of more interesting solutions, we disallow this behaviour, unless the original example itself starts or ends with that phrase.

## 4 Experimental setup

### 4.1 Datasets

The experiments have been carried out on two English sentiment analysis datasets, each consisting of sentences or short text fragments. The first is the Rotten Tomatoes dataset (Pang and Lee, 2005) which contains extracts of movie reviews with sentiment labelled as either positive or negative. We have used the predefined training, test and validation splits. The second is the Financial PhraseBank dataset (Malo et al., 2014) which contains financial news fragments with sentiment labelled as positive, neutral or negative. We have used the dataset version with at least 50% annotator label agreement, and randomly selected 10% of the data as the validation set and 10% as the test set.

For both datasets, we have excluded the training examples that the victim model classified incorrectly, as they could be said to be already "adversarial". We have also only included examples with 32 tokens or fewer, since the pre-trained paraphrase model had been trained on sequences in that range.

### 4.2 Hyperparameters and design choices

Since design choices significantly impact the attack success rate of the trained model, in the experiments we have explored the impact of two: the decoding sampling temperature used during training, which controls the exploration of the agent; and the decoding method used for inference and evaluation, which affects the diversity and quality of the generated candidate set. All other hyperparameters have been kept constant. Appendix A provides a complete list of the hyperparameters and more training details.

#### 4.2.1 Decoding temperature during training

During training, we generate the paraphrases using nucleus sampling, with the probabilities returned by a softmax operator. The temperature parameter used in the softmax, which we denote as $\tau$, visibly affects the generated text: higher temperatures produce more randomness and more diverse training examples, but also a lower percentage of valid English sentences. We have therefore investigated two values of $\tau$: a low-temperature condition of $\tau = 0.85$, and a high-temperature condition of $\tau =$

| | *Rotten Tomatoes* | | | *Financial PhraseBank* | | |
|---|---|---|---|---|---|---|
| **Decoding method** | *Untrained* | $\tau = 0.85$ | $\tau = 1.15$ | *Untrained* | $\tau = 0.85$ | $\tau = 1.15$ |
| *Sampling* | 29.3 | 38.1 | 38.3 | 19.1 | 66.0 | 62.7 |
| *Beam search* | 14.5 | **61.9** | 60.8 | 11.3 | **82.0** | 67.7 |
| *Low-diversity beam search* | 20.6 | 39.9 | 52.6 | 13.2 | 79.2 | 79.0 |
| *High-diversity beam search* | 24.5 | 52.9 | 37.4 | 21.4 | 43.4 | 59.1 |

Table 1: Attack success rate (as a percentage) of the untrained and trained models on the test set, averaged across three different random seeds. We use $\tau$ to refer to the decoding temperature during training. The best results for each dataset are in bold. Training has improved the attack success rate for all conditions ($p < 0.01$ according to a bootstrap hypothesis test (Dror et al., 2018)). Among the evaluation decoding methods, beam search had the highest success rates, followed by low-diversity beam search.

| | *Rotten Tomatoes* | | *Financial PhraseBank* | |
|---|---|---|---|---|
| **Attack method** | Success % | Avg Queries | Success % | Avg Queries |
| *Adversarial attacks (see Appendix C)* | | | | |
| LM-WR-BS-m5bw2 | 39.6 | 42 | 39.0 | 58 |
| TextFooler (Jin et al., 2019) | 67.7 | 52 | 50.9 | 72 |
| BAE-R (Garg and Ramakrishnan, 2020) | 65.2 | 53 | 60.4 | 76 |
| LM-WR-BS-m25bw5 | 69.4 | 282 | 69.2 | 290 |
| CF-WR-BS-m25bw5 | 79.4 | 327 | 64.8 | 498 |
| LM-WR-GA-p60mi20mr5 | 69.1 | 425 | 69.2 | 463 |
| LM-WR-BS-m50bw10 | 77.2 | 790 | 75.5 | 697 |
| IGA (Wang et al., 2021) | 86.9 | 512 | 76.1 | 860 |
| LM-WADR-BS-m25bw5 | 93.9 | 976 | 91.2 | 1331 |
| *Trained model* | | Queries (Avg Successes) | | Queries (Avg Successes) |
| Eval: beam search, train: $\tau = 1.15$, best run | 85.5 | 48 (19.6) | 88.1 | 48 (27.6) |

Table 2: Comparison of the best-performing trained models against a range of other adversarial attacks. For these attacks we show the average number of victim model queries needed to find a single adversarial example. In contrast, the trained model performs a fixed number of victim model's queries (48) and generates multiple adversarial examples. The rows are sorted by increasing Avg Queries value for the Rotten Tomatoes dataset.

1.15. We found both these values retain sentence semantics and give somewhat diverse examples, while also being fittingly different from each other.

#### 4.2.2 Decoding method during evaluation

During evaluation, i.e. both validation and test, we generate a set of $n = 48$ paraphrases per original example, and then check if any are valid adversarial examples based on the constraints. The decoding method used influences the characteristics of the generated text — and consequently, the attack success rate. This is a key design choice and we have therefore investigated four different methods:

**Sampling.** We have used nucleus sampling, with top-p at 0.95 and the temperature at 1.

**Beam search.** We have set the number of beams to 48, one per generated example.

**Low-diversity beam search.** Diverse beam search (Vijayakumar et al., 2018) is a beam-search variant that increases diversity of generated sequences by dividing the beams into groups and encouraging diversity between them. For this condition we have used six beam groups, set the diversity penalty to 1, and again used 48 beams, one per generated example.

**High-diversity beam search.** As above, but with 48 beam groups instead of six.

## 5 Results

### 5.1 Attack success rate

First, we have investigated if the proposed approach improved the attack success rate. As mentioned in Section 3.1, an attack is counted as successful if at least one example in the candidate set is a true adversarial example, in that it meets the validity constraints while also inducing a misclassification (NB: typically, many more than one are). The attack success rate is then computed as the percentage of successful attacks. In this section, we report the attack success rate for the trained and untrained paraphrase models, averaged across three seeds, on the test split of the two datasets, across the two temperature settings and the four decoding methods.

**Results.** The results are reported in Table 1, showing that the proposed approach has improved the attack success rate across all training condi-

tions. All improvements have been statistically significant ($p < 0.01$) according to a bootstrap test, as recommended by Dror et al. (2018). We have found no clear best between the two temperature values ($\tau = 0.85$ and $\tau = 1.15$), but beam search as the decoding method has reported the highest success rates on both datasets (61.9% on Rotten Tomatoes and 82.0% on Financial PhraseBank).

### 5.2 Comparison with established adversarial attacks

Next, we have compared the proposed approach to results from a range of adversarial attacks. We have created six different attacks with the TextAttack library (Morris et al., 2020b), varying the transformations and the search method, and we also compared against three established adversarial attack schemes: TextFooler (Jin et al., 2020), BAE-R (Garg and Ramakrishnan, 2020) and IGA (Wang et al., 2021). These attacks span a range of query budgets (i.e., the number of victim model queries per example attacked) and have a corresponding range of attack success rates (Yoo et al., 2020). To ensure a fair comparison, each attack abides by the same constraints as the trained model, while during the iterations the search is not allowed to modify the same word twice or stopwords. (more details are provided in Appendix C). For performance comparison, we have used the best-performing trained model from Table 1 for each dataset and across seeds, and we have compared it with these attacks in terms of both attack success rate and number of victim model queries.

**Results.** The results in Table 2 show that the trained model has achieved a much higher attack success rate than the corresponding adversarial attacks for a comparable average number of queries (e.g., 85.5% with 48 queries vs 67.7% with 52 for Rotten Tomatoes). In fact, its attack success rate has been similar to that of the most query-expensive attack tested, despite requiring a fraction of its queries (48 vs 976 on average for Rotten Tomatoes and 1331 for Financial PhraseBank). More so, the trained model has been able to generate not one, but many successful adversarial examples per original, averaging 19.6 out of 48 for Rotten Tomatoes and 27.6 for Financial PhraseBank. This staggering difference directly stems from the inherent design advantages of the seq2seq approach over the token-modification approach (which all these attacks use).

### 5.3 Human validation of label invariance

We have also performed a small-scale human validation of the label invariance. The assessment has been performed using samples from the Financial PhraseBank test set and comparing adversarial examples from the trained model, the untrained model, and the most successful adversarial attack we compared against. The results have showed that the trained model and the compared attack have been able to retain the same ground-truth label as the original example at approximately similar rates (59% and 50%, respectively). The untrained model has been able to retain the ground-truth label in all cases, yet at the price of a drastically lower success rate. Note that, even if discounted by these empirical label-invariance rates, the success rate of the trained model would still remain more than double that of the untrained model. Appendix D presents the full details of this validation.

## 6 Discussion

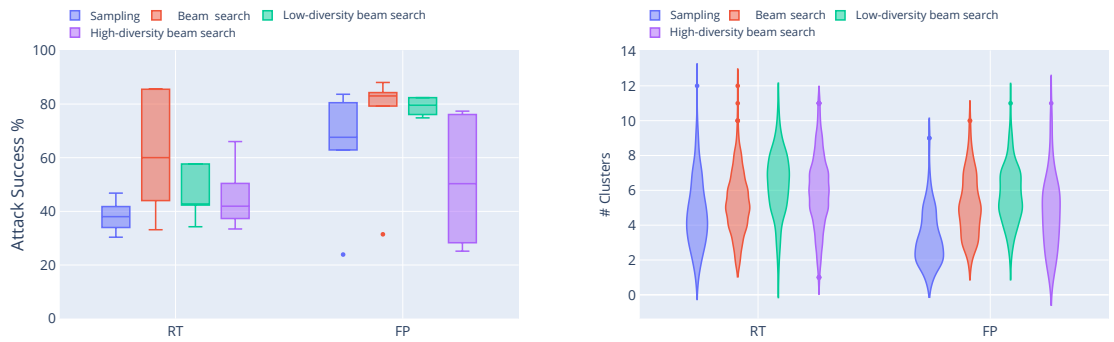
### 6.1 Impact of the decoding method

We have compared the different decoding methods used during evaluation/inference in terms of attack success rates and a diversity score. To this aim, we have defined an ad-hoc diversity score as the number of clusters returned by a clustering algorithm (HDBSCAN (McInnes et al., 2017)) over the generated candidate sets (full details are available in Appendix E).

Figure 4a shows that in terms of attack success rate, beam search has been the best, followed by low-diversity beam search, although we observed significant variation between runs for all methods. In turn, Figure 4b shows that in terms of the diversity score, low-diversity beam search has generated the most diverse examples, while sampling has generated the least. Appendix E also shows a comparison of fluency and diversity of generated bi-grams. Overall, it could be argued that low-diversity beam search has proved the best decoding method for this task in terms of success attack rate/diversity trade-off.

### 6.2 Learned strategies

During training, the model has displayed a wide range of different generating behaviours, such as phrase shuffling, synonym swapping and double negatives, a selection of which is shown in Table 3. Some behaviours have proved compatible with our validity constraints, others have violated them. In




(a) (b)

Figure 4: Attack success rate and diversity of decoding methods. For each graph: RT = Rotten Tomatoes, FP = Financial PhraseBank. (a) Attack success rate by decoding evaluation method. Beam search and low-diversity beam search perform best. (b) Candidate set diversity of each decoding method, which we measure using a cluster-based score (see Appendix E). More clusters means a more diverse candidate set.

| **Transformation** | **Example** |
|---|---|
| Original | safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye |
| Genuine paraphrase | safe conduct might be ambitious and well-intentioned, but it misses the entertainment bull's-eye |
| Synonym swapping | safe conduct, however ambitious and well-intentioned, fails to strike the entertainment bull's-eye |
| Case changes | safe Conduct, however Ambitious And well-intentioned, fails to hit the entertainment bull's-eye |
| Adding/removing punctuation | safe conduct however ambitious and well-intentioned, fails to hit the entertainment bullseye..:;: |
| Ignoring grammar | safe conduct ambitious well-intentioned bull's-eye |
| Phrase shuffling | safe conduct, fails to hit the entertainment bull's-eye, however ambitious and well-intentioned |
| Contradictions | safe conduct, however ambitious and well-intentioned, hit the entertainment bull's-eye |
| Sentence truncation | safe conduct, however ambitious and well-intentioned |
| Padding to max length | safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye and and and and and and |
| Very short sentences | safe conduct. |
| Inserting phone-numbers | safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye 888-739-5110 888-739-5110 |
| Using Unicode characters | safe cond©√t, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye. |
| Adding linking contrast phrases | although safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye, nonetheless |
| Repeating phrases | safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye, entertainment bull's-eye |
| Change language of fragment | safe conduct, however ambitieux et well-intentioned, fails to hit the entertainment bull's-eye |
| Rhetorical question | safe conduct, however ambitious and well-intentioned, fails to hit the entertainment bull's-eye - but why? |
| Double negatives | safe conduct, however ambitious and well-intentioned, fails to fail to hit the entertainment bull's-eye |

Table 3: Examples of the model's generating behaviours during training. Rather than only paraphrasing, the model has exhibited a variety of different behaviours. We have introduced two additional constraints to disallow the behaviours of generating very short sentences and adding linking contrast phrases, but we have allowed the others.

the initial stages of our research, we had to adjust the reward function repeatedly to disallow some unwanted behaviours, leading to the additional constraints of Section 3.5.[4] Some behaviours, such as ignoring grammar, can be considered exploitations of the individual components of the reward function (i.e., "reward hacking"). In addition, the constraints have not been able to perfectly filter all the actual adversarial examples. However, since the components and the constraints are simply proxies for human preferences, the reward can still be effective so long that these behaviours remain limited.

## 7 Conclusion

This paper has proposed an approach for generating adversarial attacks for a text classifier based on fine-tuning a seq2seq paraphrase model. The proposed approach has trained the paraphrase model with an original reward function that encourages misclassifications in the victim model while simultaneously ensuring that the generated attacks abide by a set of validity constraints. The experimental results over two datasets have shown that the trained model has been able to produce many more adversarial examples than the untrained model. It has also proved much more efficient than comparable token-modification attacks in terms of the success rate/number of queries trade-off. Future work could include exploring more efficient and stable training algorithms, incorporating actual human preferences into the reward objective, and experimenting with different pre-trained seq2seq models, such as those for style transfer or dialogue generation.

[4]This analysis was carried out solely on the training set.

## 8 Ethical considerations

In this paper we have trained a model to attack text classifiers. The obvious danger is that the proposed approach could be used by an actual adversary to attack real-world models. The general justification for adversarial attack research is that the preventative identification of effective attacks can help identify vulnerabilities and develop defences. In addition, research on adversarial attacks can contribute to data augmentation for robust model training.

In terms of biases, the trained model will likely reflect both the biases of the original examples used for its input, and those of the baseline paraphraser it was built upon. However, this seems to be in common with all other conditional text generators such as machine translation and summarisation models. No explicit mitigation has been put in place for this.

## References


Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3601–3608.

Prithiviraj Damodaran. 2021. Parrot: Paraphrase generation for nlu.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2014. HotFlip: White-Box Adversarial Examples for Text Classification.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075, Florence, Italy. Association for Computational Linguistics.

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 2020, Online, November 16-20, 2020*, pages 6174–6181. Association for Computational Linguistics.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. 2018. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018*.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. 2018. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E. Turner, and Douglas Eck. 2017. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 1645–1654. PMLR.

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

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 8018–8025.

Solomon Kullback and R. A. Leibler. 1951. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86.

Jiaying Lu, Xin Ye, Yi Ren, and Yezhou Yang. 2022. Good, better, best: Textual distractors generation for multi-choice VQA via policy gradient. In *CVPR 2022 Workshop on Open-Domain Retrieval Under a Multi-Modal Setting*, O-DRUM @ CVPR 2022.

P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11).

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Paul Michel, Xian Li, Graham Neubig, and Juan Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. 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 3103–3114.

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

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020b. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Proceedings of HLT-NAACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4569–4580, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Yankun Ren, Jianbin Lin, Siliang Tang, Jun Zhou, Shuang Yang, Yuan Qi, and Xiang Ren. 2020. Generating natural language adversarial examples on a large scale with generative models. *CoRR*, abs/2003.10388.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. 2021. Token-modification adversarial attacks for natural language processing: A survey.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA. MIT Press.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Prashanth Vijayaraghavan and Deb Roy. 2019. Generating black-box adversarial examples for text classifiers using a deep reinforced model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 711–726. Springer.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language*

*Processing (EMNLP)*, pages 5531–5546, Online. Association for Computational Linguistics.

Ke Wang and Xiaojun Wan. 2018. Sentigan: Generating sentimental texts via mixture adversarial networks. pages 4446–4452.

Xiaosen Wang, Jin Hao, Yichen Yang, and Kun He. 2021. Natural language adversarial defense through synonym encoding. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pages 823–833. AUAI Press.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

Catherine Wong. 2017. Dancin seq2seq: Fooling text classifiers with adversarial text example generation.

Jin Yong Yoo, John Morris, Eli Lifland, and Yanjun Qi. 2020. Searching for a search method: Benchmarking search algorithms for generating NLP adversarial examples. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 323–332, Online. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial Attacks on Deep-learning Models in Natural Language Processing. *ACM Transactions on Intelligent Systems and Technology*, 11(3):1–41.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

## A Training details

This section provides all the details of our training setup.

We have used an AdamW optimiser with early stopping, using the attack success rate of the validation set as the metric, and stopping once the metric dropped below the running median, or when a maximum number of epochs was reached (100 for Rotten Tomatoes, 200 for Financial PhraseBank). We have not done any layer freezing during training as we noticed that it tended to reduce performance.

After the sample selection from the datasets, the splits contained the following number of examples: Rotten Tomatoes — training: 2972, validation: 367, test: 359; Financial PhraseBank — training: 1370, validation: 167, test: 159. The hyperparameters that have been kept constant across all experiments are listed in Table 4. The details of the various models — paraphrase model, victim models, and reward component models — are given in Table 5.

| **Hyperparameter** | **Value** |
|---|---|
| *General* | |
| LR | $1 \times 10^{-4}$ |
| Batch size | 32 |
| Gradient accumulation steps | 2 |
| Max paraphrase length | 48 |
| Min paraphrase length | 3 |
| Max original length | 32 |
| Padding multiple | 8 |
| *Training generation* | |
| Generated sequences per original | 1 |
| Top-p | 0.95 |
| *Eval generation* | |
| Generated sequences per original | 48 |
| Top-p (S) | 0.95 |
| Temperature (S) | 1 |
| Number of beams (BS, DBS) | 48 |
| Diversity penalty (DBS) | 1 |
| *Reward function* | |
| Reward bounds | [0, $\alpha$ = 10] |
| Victim degradation multiplier ($\eta$) | 35 |
| $D_{KL}$ scaling coefficient ($\beta$) | 0.4 |
| Character difference threshold | $\pm$30 |
| Cosine similarity threshold | 0.8 ($\geq$) |
| NLI contradiction threshold | 0.2 ($\leq$) |
| Linguistic acceptability threshold | 0.5 ($\geq$) |

Table 4: Hyperparameters used for training and evaluation across all experiments. S refers to sampling, BS to beam search, DBS to the two diverse beam search conditions, and NLI to natural language inference. The signs for the thresholds indicate the direction needed to meet the condition.

The GPU model used for training has been an NVIDIA Quadro RTX 6000 with 24GB RAM. Each training run has used a single GPU and the runtime varied from around 4 hours to around 64. The runtime depended heavily on the decoding method (*sampling* was the fastest) and the maximum number of epochs. The training could be potentially sped up by reducing the number of the generated paraphrases during validation. In addition, using a GPU with more memory would allow increasing the batch size, which would also reduce the training time considerably. Occasionally, for some random seeds and conditions, we have run out of memory on the GPU, most likely due to unusually long predictions; repeating the runs with different seeds has always circumvented this issue.

## B Limitations

The proposed approach is certainly not without limitations. The training process is GPU-intensive, and a training run can take from a few hours to a few days on a single GPU. A contemporary GPU with at least 16 GB of RAM is probably the minimum computing requirement, with more RAM allowing larger batch sizes and training speed-ups. In addition, during training the model must query the victim model many times, which may not be possible under many scenarios. We have also observed a high variance in the results across random seeds. This problem is well known in reinforcement learning (Henderson et al., 2018) and may be mitigated by more sophisticated baselines, such as RELAX (Grathwohl et al., 2018). Lastly, the proposed approach seems to work best for short sentences, as the paraphrase quality starts to visibly drop as the sentence length increases. However, this may not be a major limitation for the generation of adversarial attacks since larger blocks of text could always be subdivided.

As a final note, the experiments have only covered two datasets in English and have not investigated other languages.

## C Token-modification attacks

We have created a variety of attack types using the TextAttack library of (Morris et al., 2020b). For all attacks, we have set the goal function as untargeted classification (i.e., attacking the correct label with any other class). We have varied the transformation, the search method, and some hyperparameters of the algorithms, and used the sets

| Purpose | Size (MB) | Identifier |
|---|---|---|
| Paraphraser | 892 | prithivida/parrot_paraphraser_on_T5 (Damodaran, 2021) |
| Victim model (RT) | 268 | textattack/distilbert-base-uncased-rotten-tomatoes |
| Victim model (FP) | 329 | mrm8488/distilroberta-fine-tuned-financial-news-sentiment-analysis |
| Acceptability | 47 | textattack/albert-base-v2-CoLA |
| STS | 134 | sentence-transformers/paraphrase-MiniLM-L12-v2 |
| Contradiction | 54 | howey/electra-small-mnli |

Table 5: The models used in this paper. We used small, distilled models to increase training speed and because of our GPU memory requirements, but larger models would give better performance. As above, RT stands for Rotten Tomatoes, FP for Financial PhraseBank. All identifiers refer to models on the Hugging Face Model Hub.

of constraints of Section 3.5. We have also used two extra constraints commonly used to improve the effectiveness of the search (e.g. (Cheng et al., 2020; Jin et al., 2019)): the attacks have not been allowed to modify the same word twice, nor modify stopwords. Tables 5 and 6 provide full details.

# D Human validation of the adversarial examples

This section provides full details of the human validation of the adversarial examples. The validation has used three annotators, one of which was a native English speaker and two were proficient English second-language speakers[5].

## D.1 Evaluated approaches

We have compared the untrained paraphrase model, the trained model, and the best-performing token-modification algorithm (LM-WADR-BS-bw5m25 from Table 2) based on the attack success rate. For the trained model we have chosen a run using low-diversity beam search, since in Section 6.1 we have argued that this has proved the best decoding method overall.

## D.2 Evaluated examples

We have used 41 original examples from the Financial PhraseBank test set and generated adversarial examples for them. The original examples were split into two groups. The first is where all the three methods were able to produce adversarial examples (18 examples). In the second, the trained model and the token-modification algorithm were able to produce adversarial examples, but the untrained paraphrase model was not (23 examples). The original examples have been selected so as to have a close-to-balanced class distribution.

[5]And, for full disclosure, co-authors of this submission. For this reason, the validation has been carried out *blindly*, without knowledge of the generating approach of the examples under assessment.

## D.3 Filtering adversarial candidates

Both the untrained paraphrase model and the trained model can generate up to 48 adversarial examples per original example, which are too many for a human annotator to easily evaluate. To amend this, we have filtered the set of generated adversarial examples to get a smaller, selected subset that is less onerous to evaluate. The pseudocode for the filtering is provided in Algorithm 1. The goal is to end up with a set of roughly 6-12 candidates per original example that approximately preserve the diversity of the larger set. In brief, the algorithm computes sentence embeddings for all candidates, applies a dimensionality reduction algorithm, UMAP (McInnes et al., 2018), and then applies a hierarchical clustering algorithm, HDBSCAN (McInnes et al., 2017). We have eventually sampled a few candidates from each cluster, with the actual number depending on the number of clusters extracted by HDBSCAN.

**Algorithm 1** Pseudocode for filtering the generated adversarial examples.

```
Select all successes 𝕊 from candidate set ℂ ∈ 𝕊
if |𝕊| ≤ 6 then
    Return 𝕊′ = 𝕊
else
    Compute sentence embeddings E(s), ∀s ∈ 𝕊
    Reduce dimensionality to get E′(s), ∀s ∈ 𝕊
    Cluster E′(s)
    Create 𝕊′ by sampling from each cluster
    Return 𝕊′
end if
```

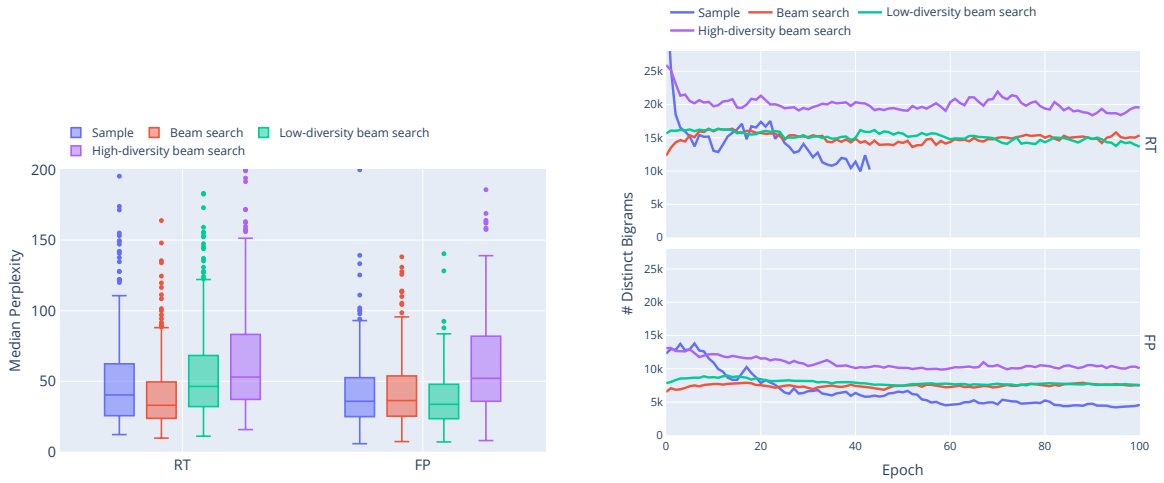
## D.4 Label invariance

The aim of the validation has been to assess whether the adversarial examples generated by the three approaches had retained the same ground-truth label as the original, and this question was

| Attack identifier | Transformations | Search method | Max candidates | Parameters |
|---|---|---|---|---|
| LM-WR-BS-m5bw2 | Word replacement using language model | Beam search | 5 | $bw = 2$ |
| LM-WR-BS-m25bw5 | Word replacement using language model | Beam search | 25 | $bw = 5$ |
| CF-WR-BS-m25bw5 | Word replacement using CF embedding | Beam search | 25 | $bw = 5$ |
| LM-WR-GA-p60mi20mr5 | Word replacement using language model | Genetic algorithm | 25 | $p = 60, mi = 20, mr = 5$ |
| LM-WR-BS-m50bw10 | Word replacement using language model | Beam search | 50 | $bw = 10$ |
| LM-WADR-BS-m25bw5 | Word replacement, addition, deletion using language model | Beam search | 25 | $bw = 5$ |
| TextFooler | Word replacement using CF embedding | WIR (delete) | 50 | - |
| BAE-R | Word replacement using language model | WIR (delete) | 50 | - |
| IGA | Word replacement using CF embedding | Genetic algorithm | 50 | $p = 60, mi = 20, mr = 5$ |

Table 6: Token modification attacks and their parameters. All attacks were generated using the TextAttack package. We have used a DistilRoBERTa language model*. CF embedding refers to GloVE word embeddings that have been counter-fitted, a procedure introduced in Mrkšić et al. (2016) and commonly used in finding word replacements. WIR (delete) stands for Word Importance Ranking by deletion, a search method where the importance of each token is estimated by deleting it and measuring the true-class confidence drop in the victim model. We use $bw$ to mean beam width, and for the genetic algorithm search parameters, $p$ is the population size, $mi$ the maximum number of iterations, and $mr$ the maximum number of replacements per index.

* https://huggingface.co/distilroberta-base




(a) (b)

Figure 5: Fluency scores for the various decoding methods. RT = Rotten Tomatoes, FP = Financial PhraseBank. (a) Median perplexity of the generated candidate sets, with examples combined from the top two runs of each decoding method. Three of the methods have been approximately comparable, while high-diversity beam search has consistently produced the least fluent candidates. (b) Average number of distinct bigrams generated per epoch performing evaluation on the training set. High-diversity beam search (in purple) has consistently generated more unique bigrams than the other methods. The *sampling* decoding method has displayed a marked decrease in diversity along the epochs, while the others have remained approximately constant. These results confirm the expected trade-off between fluency and diversity.

asked to the annotators (Figure 6 shows a screenshot of the instructions). To determine the sentiment, we have used the same instructions as the original annotation of the Financial PhraseBank dataset (Malo et al., 2014). We have used majority voting among the annotators to determine the label invariance.

The results of this assessment have been: 100% of the adversarial examples generated by the original paraphrase model have retained their label; 59% for the trained model; and 50% for the token-modification algorithm. This shows that the trained model has degraded paraphrasing capability compared to the original model, but it still has achieved higher label invariance than the token-modification approach. These results should be judged alongside the success rate, which has been approximately 4 times higher for the trained model than the original model on the Financial PhraseBank test set (Table 1). It is also worth noting that label invariance rates vary considerably in the literature, foremost because there are no "standard settings" across papers. The rates depend on many factors, such as the instructions given to the evaluators, their harshness

or lenience, the use of crowd-sourcing platforms, the datasets used, etc. The most important requirement is that the evaluation is applied equally to all baselines, which we do here.

## E Analysis of the decoding methods

This section describes the analysis of the four decoding methods and shows further results on fluency and diversity.

**Preprocessing.** To measure diversity and fluency, we have selected the two runs with the highest attack success rate per decoding condition across both training temperatures.[6]

**Diversity.** To quantify the diversity, we have built a score using the same procedure used for the selection of the adversarial examples for the human validation, described in full in Appendix D.3. The score has been simply defined as the number of distinct clusters returned by HDBSCAN, with the individual examples not included in any cluster counted as clusters themselves.

The results are shown in Figure 4 in the main paper. We found that the high-diversity beam search showed less diversity than the low-diversity beam search. This was because it tended to generate many "degenerate" examples that were clustered together by our clustering procedure. The sampling method showed low diversity as it generated fewer unique examples on average than the other conditions. We speculate that the token probability over the vocabulary tends to become more concentrated as training progresses, as supported by Figure 5b.

**Fluency.** As commonly done (e.g. (Wang and Wan, 2018)), we have used the language model's perplexity as a proxy for fluency. We have also used the number of unique generated bigrams performing evaluation on the training set after every epoch as a further measurement of diversity. The results are shown in Figure 5.

## F Examples of generated text

Tables 7 and 8 show examples of the generated text. We present one example per dataset, including all the decoding methods with the best untrained and trained models, and the best-performing token-modification attack. For reasons of space, the examples only show eight generated adversarial candidates for each approach.

[6] We select the best two because there is significant variation in runs between random seeds and it would not be beneficial to compare models that have failed to train properly.

Each decoding method has exhibited specific trends. Sampling and beam search have generated examples that are very similar to each other, only differing slightly. Additionally, sampling has generated fewer unique candidates than the other methods. Both low and high-diversity beam search have generated good quality candidates, but also many that were ungrammatical or incoherent; and more so for the high-diversity case. Overall, we notice that the generated adversarial candidates seem to frequently contain consistent phrases or terms, which makes them *universal adversarial examples* (Wallace et al., 2019). The training procedure clearly learns which phrases affect sentiment across many examples, and the approach inserts them into its generated solutions. This is less desirable than a model that can generate "tailored" paraphrases for each original example. The paraphrasing capability of the trained model is still remarkable, although at times visibly lower than that of the untrained model. Overall, the trained model seems to have been able to generate better paraphrases on Financial PhraseBank than Rotten Tomatoes, likely because of its simpler language.

| Approach | Generated Text |
|---|---|
| Original example | suffers from unlikable characters and a self-conscious sense of its own quirky hipness. |
| Token-modification | suffers from unlikable characters and gains a self-conscious sense of its own quirky hipness . |
| Sampling, untrained | suffers from unlikable characters and has a sense of self-conscious hipness.<br>suffers from unlikable characters and a self-conscious sense of its own quirky hipness.<br>suffers from unlikable characters and a self-conscious sense of its own quirky hipness..<br>suffers from unlikable characters and an in-your-face sense of quirky hipness.<br>suffers from unlikable characters and a self-conscious sense of its own quirky hipness...<br>suffers from unlikable characters and an internal sense of eccentric hipness..<br>suffers from unlikeable characters and a self-conscious sense of its own quirky hipness. ”<br>suffers from unlikable characters and self-conscious sense of humour. |
| Sampling, trained | Unlikable characters may even have a self-conscious hipness<br>Unlikable characters may even have a self-conscious sense of their own quirky hip<br>Unlikable characters may even have self-conscious hipness<br>Unlikable characters may even have a self-conscious sense of their own hip<br>Unlikable characters might even have an self-conscious hip hop habit<br>Unlikable characters might even have a self-conscious hip hop attitude<br>Unlikable characters might even have a self-conscious sense of their own quirky hip<br>Unlikable characters may even have self-conscious hip-ness |
| Beam search, untrained | it suffers from unlikable characters and a self-conscious sense of its own quirky hipness.<br>suffers from unlikable characters and a self-conscious sense of its own quirky hip-hop.<br>suffers from unlikable characters and a self-conscious sense of its own quirky hip-ness.<br>suffers from unlikable characters and self-conscious sense of its own quirky hipness.<br>he suffers from unlikable characters and a self-conscious sense of his own quirky hipness.<br>suffers from unlikable characters and a sense of self-consciousness of its own quirky hipness..<br>suffers from unlikable characters and a self-conscious sense of quirky hipness..<br>suffers from unremarkable characters and a self-conscious sense of its own quirky hipness. |
| Beam search, trained | It suffers from unlikable characters and a self-conscious sense of its own quirky hipness but there is something<br>It suffers from unlovable characters and a sense of its own quirky hipness but there is something<br>It suffers from unlikable characters and an unconscious sense of its own quirky hipness but there is something<br>It suffers from unlikable characters and an inner sense of its own quirky hipness but there is something<br>It suffers from unlikable characters that have a sense of their own quirky hipness but there is something<br>It suffers from an unlikable character and a sense of its own quirky hipness but there is something<br>It suffers from unlikable characters and a feeling of its own quirky hipness but there is something<br>It suffers from unlikable characters and a sense of its own quirky hipness but there is some |
| Low-diversity beam search, untrained | suffers from unlikable characters and a self-conscious sense of its own quirky hipness.<br>suffers from unlikable characters and a self-conscious sense of its own quirky hipness. ”<br>suffers from unlikable characters and a self-conscious sense of its own quirky hipness.<br>” suffers from unlikable characters and a self-conscious sense of its own quirky hipness<br>a character that suffers from an unlikable character and a self-conscious sense of its own quirky hipness<br>there is a lack of likeable characters and a fear of its own eccentricity.<br>there is a lack of likeable characters and a fear of its own quirky hips.<br>there is a lack of likeable characters and a sense of its own quirky hipness.. |
| Low-diversity beam search, trained | It suffers from unlikable characters but uses a sense of its own quirky hipness<br>It suffers from unlikable characters but includes a sense of its own quirky hip<br>It suffers from unlikable characters but often displays its own quirky hip<br>It comes from unlikable characters but uses a sense of its own quirky hipness<br>It comes from unlikable characters but includes a sense of its own quirky hipness<br>It struggles with unlikable characters but often displays its own quirky hip<br>It also has unlikable characters but is intended for its own quirky hip<br>It struggles with unlikable characters but often displays its own quirky hipness but uses some hip |
| High-diversity beam search, untrained | has unlikable characters and a sense of its own quirky hipness.<br>it has unlikable characters and a self-conscious sense of its own quirky hipness.<br>suffers from unlikable characters and an uncanny sense of its own quirky hipness.<br>it has unlikable characters and a sense of its own quirky hipness.<br>the characters are not likable but the sense of their own quirky hipness is a problem.<br>the characters are not likable but the sense of their own quirky hipness is a problem.<br>has unlikable characters and a self-conscious sense of its own quirky hipness....<br>shows an unlikely character with an unrequited feeling of hipster... |
| High-diversity beam search, trained | The unlikable characters should have a self-conscious sense of hipness<br>Sometimes it should suffer from unlikable characters but sometimes self-conscious senses of hipness<br>Unlikable characters should have a self-conscious sense of hipness<br>This can involve unlikable characters but sometimes self-conscious hipness.<br>It must suffer from unlikeable characters but a self-conscious sense of hipness<br>The unlikeable characters must have a self-conscious sense of hipness<br>The unlikeable character should have a self-conscious sense of its own hipness<br>it will have to suffer from unlikable characters but often quirky hipness |

Table 7: Examples of generated adversarial candidates for the various approaches for an original example from the Rotten Tomatoes dataset. The original example is in black and has negative sentiment. We use light grey for unsuccessful candidates and green for a successful adversarial example that changes sentiment to positive.

| Approach | Generated Text |
|---|---|
| Original example | In addition , the company will reduce a maximum of ten jobs . |
| Token-modification | In addition , the company will reduce a maximum requirement of ten jobs . |
| Sampling, untrained | In addition, the company can reduce the jobs of up to ten people.<br>In addition the company will cut a maximum of ten jobs.<br>In addition he says the company will close a maximum of ten jobs.<br>In addition, the company will eliminate up to ten jobs.<br>In addition the company will reduce up to ten jobs.<br>In addition the company has cut a maximum of ten jobs.<br>In addition " the company will cut 10 jobs in 10 months.<br>In addition to the job cuts the company will limit itself to ten jobs. |
| Sampling, trained | In addition, the company has succeeded in reducing a maximum of ten jobs.<br>In addition, the company has succeeded in decreasing a maximum of ten jobs.<br>In addition, the company has succeeded in cutting a maximum of ten jobs.<br>In addition, the company has succeeded inreducing a maximum of ten jobs. |
| Beam search, untrained | In addition, the company will reduce a maximum of ten jobs.<br>In addition he will reduce a maximum of ten jobs.<br>In addition, the company will reduce a maximum of ten jobs.<br>The company will also reduce a maximum of ten jobs.<br>In addition, the company will cut a maximum of ten jobs. "<br>In addition the company will cut up to ten jobs.<br>In addition, the company will eliminate a maximum of ten jobs.<br>In addition the company will cut a maximum of ten jobs.. |
| Beam search, trained | In addition, the company will also be successful in reducing a maximum of 10 jobs<br>In addition, the company will also see success in reducing a maximum of 10 jobs<br>In addition, the company will also see success in reducing a maximum of 10 jobs.<br>In addition, the company will also see success in reducing ten jobs.<br>In addition, the company will also see success at reducing a maximum of 10 jobs.<br>In addition, the company will also have success in cutting a maximum of 10 jobs.<br>In addition, the company will also find success in reducing a maximum of 10 jobs<br>In addition, the company will also find success in reducing a maximum of 10 jobs. |
| Low-diversity beam search, untrained | In addition, the company will reduce a maximum of ten jobs.<br>In addition, the company will reduce a maximum of ten jobs..<br>In addition, the company will reduce a maximum of ten jobs. "<br>In addition the company will cut a maximum of ten jobs. "<br>In addition, the company will reduce a maximum of ten jobs...<br>In addition, the company will reduce a maximum of ten jobs in the region.<br>He said: In addition he would reduce the company's total number of jobs to 10.<br>In addition, the company will reduce a maximum of ten job cuts. |
| Low-diversity beam search, trained | In addition, the company will benefit from reducere ten jobs.<br>This is good for decreasing a maximum of 10 jobs.<br>Along with this reduction of 10 jobs, the company will benefit from reductions.<br>The company will benefit from r$\surd$©duiing a maximum of 10-jobs.<br>In addition, the company will benefit from reductions for ten jobs.<br>In addition the company will benefit from reductions for the maximum of 10 jobs.<br>As a result the company will benefit from reductions of up to 10 jobs.<br>The company will benefit from reducing ten jobs. |
| High-diversity beam search, untrained | The company will also reduce a maximum of ten jobs.<br>A further reduction of ten jobs is planned. "<br>A maximum of ten jobs will be lost in the company.<br>Furthermore, the company will reduce up to ten jobs..<br>The company will reduce its workforce by up to 10 percent. "<br>It also plans to eliminate at least ten jobs in the company..<br>Further the company will reduce a maximum of 10 jobs.<br>Additionally the company will lose up to ten jobs. |
| High-diversity beam search, trained | The company also reduced the reduction of a maximum of ten jobs.<br>Furthermore the company will reduce the reduction of ten jobs.<br>This also improved reduction of up to ten jobs.<br>The company also reduced the reduction of ten jobs..<br>the company will also improve reductions for up to ten jobs.<br>Also the company would reduce a maximum of ten jobs.<br>The company also reduced the reduction of ten jobs...<br>Additionally the firm would also improve the reduction of the maximum of 10 jobs. |

Table 8: Examples of generated adversarial candidates for the various approaches for an original example from the Financial PhraseBank dataset. The original example is in black and has negative sentiment. We use light grey for unsuccessful candidates, and green and blue for successful adversarial examples that changed the sentiment to positive and neutral, respectively. (NB: the "Sampling, trained" approach only produced four unique candidates for this example.)

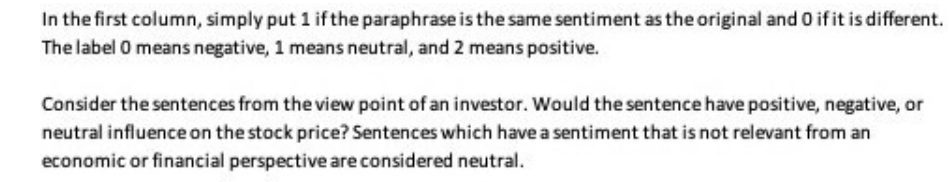



Figure 6: Screenshot of the instructions provided to the annotators for the human validation.