Universal Adversarial Attacks with Natural Triggers for Text Classification

Liwei Song[§] Xinwei Yu[§] Hsuan-Tung Peng[§] Karthik Narasimhan
Princeton University
{liweis, xinweiy, hsuantungp, karthikn}@princeton.edu

Abstract

Recent work has demonstrated the vulnerability of modern text classifiers to *universal adversarial attacks*, which are input-agnostic sequences of words added to text processed by classifiers. Despite being successful, the word sequences produced in such attacks are often ungrammatical and can be easily distinguished from natural text. We develop adversarial attacks that appear closer to natural English phrases and yet confuse classification systems when added to benign inputs. We leverage an adversarially regularized autoencoder (ARAE) (Zhao et al., 2018a) to generate triggers and propose a gradient-based search that aims to maximize the downstream classifier's prediction loss. Our attacks effectively reduce model accuracy on classification tasks while being less identifiable than prior models as per automatic detection metrics and human-subject studies. Our aim is to demonstrate that adversarial attacks can be made harder to detect than previously thought and to enable the development of appropriate defenses.[1]

1 Introduction

Adversarial attacks have recently been quite successful in foiling neural text classifiers (Jia and Liang, 2017; Ebrahimi et al., 2018). *Universal adversarial attacks* (Wallace et al., 2019; Behjati et al., 2019) are a sub-class of these methods where the same attack perturbation can be applied to *any input to the target classifier*. These attacks, being input-agnostic, point to more serious shortcomings in trained models since they do not require re-generation for each input. However, the attack sequences generated by these methods are often meaningless and irregular text (e.g., "zoning tapping fiennes" from Wallace et al. (2019)). While human readers can easily identify them as unnatural, one can also use simple heuristics to spot such attacks. For instance, the words in the above attack trigger have an average frequency of 14 compared to 6700 for words in benign inputs in the Stanford Sentiment Treebank (SST) (Socher et al., 2013).

In this paper, we design *natural attack triggers* by using an adversarially regularized autoencoder (ARAE) (Zhao et al., 2018a), which consists of an auto-encoder and a generative adversarial network (GAN). We develop a *gradient-based search* over the noise vector space to identify triggers with a good attack performance. Our method – Natural Universal Trigger Search (NUTS) – uses projected gradient descent with l_2 norm regularization to avoid using out-of-distribution noise vectors and maintain the naturalness of text generated.[2]

Our attacks perform quite well on two different classification tasks – sentiment analysis and natural language inference (NLI). For instance, the phrase *combined energy efficiency*, generated by our approach, results in a classification accuracy of 19.96% on negative examples on the Stanford Sentiment Treebank (Socher et al., 2013). Furthermore, we show that our attack text does better than prior approaches on three different measures – average word frequency, loss under the GPT-2 language model (Radford et al., 2019), and errors identified by two online grammar checking tools (scr; che). A human judgement study shows that up to 77% of raters find our attacks more natural than the baseline and almost 44% of humans find our attack triggers concatenated with benign inputs to be natural. This demonstrates that using techniques similar to ours, adversarial attacks could be made much harder to detect than previously thought and we require the development of appropriate defenses in the long term for securing our NLP models.

[§]Equal contribution

[1]Our code is available at `https://github.com/Hsuan-Tung/universal_attack_natural_trigger`.

[2]We define naturalness in terms of how likely a human can detect abnormalities in the generated text.

2 Related Work

Input-dependent attacks These attacks generate specific triggers for each different input to a classifier. Jia and Liang (2017) fool reading comprehension systems by adding a single distractor sentence to the input paragraph. Ebrahimi et al. (2018) replace words of benign texts with similar tokens using word embeddings. Similarly, Alzantot et al. (2018) leverage genetic algorithms to design word-replacing attacks. Zhao et al. (2018b) adversarially perturb latent embeddings and use a text generation model to perform attacks. Song et al. (2020) develop natural attacks to cause semantic collisions, i.e. make texts that are semantically unrelated judged as similar by NLP models.

Universal attacks Universal attacks are input-agnostic and hence, word-replacing and embedding-perturbing approaches are not applicable. Wallace et al. (2019) and Behjati et al. (2019) concurrently proposed to perform gradient-guided searches over the space of word embeddings to choose attack triggers. In both cases, the attack triggers are meaningless and can be easily detected by a semantic checking process. In contrast, we generate attack triggers that appear more natural and retain semantic meaning. In computer vision, GANs have been used to create universal attacks (Xiao et al., 2018; Poursaeed et al., 2018). Concurrent to our work, Atanasova et al. (2020) design label-consistent natural triggers to attack fact checking models. They first predict unigram triggers and then use a language model conditioned on the unigram to generate natural text as the final attack, while we generate the trigger directly.

3 Universal Adversarial Attacks with Natural Triggers

We build upon the universal adversarial attacks proposed by Wallace et al. (2019). To enable *natural* attack triggers, we use a generative model which produces text using a continuous vector input, and perform a gradient-guided search over this input space. The resulting trigger, which is added to benign text inputs, is optimized so as to maximally increase the loss under the target classification model.

Problem formulation Consider a pre-trained text classifier F to be attacked. Given a set of benign input sequences $\{x\}$ with the same ground truth label y, the classifier has been trained to predict $F(x) = y$. Our goal is to find a single input-agnostic trigger, t, that when concatenated[3] with *any* benign input, causes F to perform an incorrect classification, i.e., $F([t; x]) \neq y$, where ; represents concatenation. In addition, we also need to ensure the trigger t is natural fluent text.

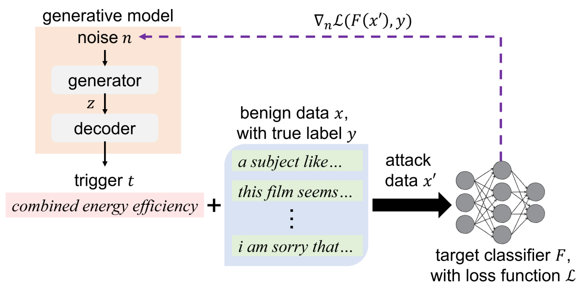
generative model
noise n
generator
z
decoder
trigger t
combined energy efficiency
+
benign data x, with true label y
a subject like...
this film seems...
i am sorry that...
attack data x'
target classifier F, with loss function $\mathcal{L}$
$\nabla_n \mathcal{L}(F(x'), y)$

Figure 1: **Overview of our attack.** Based on the gradient of the target model's loss function, we iteratively update the noise vector n with small perturbation to obtain successful and natural attack triggers.

Attack trigger generation To ensure the trigger is natural, fluent and carries semantic meaning, we use a pre-trained adversarially regularized autoencoder (ARAE) (Zhao et al., 2018a) (details in Section 4). The ARAE consists of an encoder-decoder structure and a GAN (Goodfellow et al., 2014). The input is a standard Gaussian noise vector n, which is first mapped to a latent vector z by the generator. Then the decoder uses this z to generate a sequence of words – in our case, the trigger t. This trigger is then concatenated with a set of benign texts $\{x\}$ to get full attack texts $\{x'\}$. The overall process can be formulated as follows:

$$z = \textsc{Generator}(n); t = \textsc{Decoder}(z);$$
$$x' = [t; x]$$

We then pass each x' into the target classifier and compute the gradient of the classifier's loss with respect to the noise vector, $\nabla_n \mathcal{L}(F(x'), y)$. Backpropagating through the decoder is not straightforward since it produces discrete symbols. Hence, we use a reparameterization trick similar to the trick in Gumbel softmax (Jang et al., 2017) to sample words from the output vocabulary of ARAE model as a one-hot encoding of triggers, while allowing gradient backpropagation. Figure 1 provides an overview of our attack algorithm, which we call Natural Universal Trigger Search (NUTS).

Ensuring natural triggers In the ARAE model, the original noise vector n_0 is sampled from a stan-

[3]We follow Wallace et al. (2019) in adding the triggers in front of the benign text.

dard multi-variant Gaussian distribution. While we can change this noise vector to produce different outputs, simple gradient search may veer significantly off-course and lead to bad generations. To prevent this, following Carlini and Wagner (2017), we use projected gradient descent with an l_2 norm constraint to ensure the noise n is always within a limited ball around n_0. We iteratively update n as:

$$n_{t+1} = \Pi_{\mathcal{B}_\epsilon(n_0)}[n_t + \eta \nabla_{n_t} \mathcal{L}(F(x'), y)], \quad (1)$$

where $\Pi_{\mathcal{B}_\epsilon(n_0)}$ represents the projection operator with the l_2 norm constraint $\mathcal{B}_\epsilon(n_0) = \{n \,|\, \|n - n_0\|_2 \le \epsilon\}$. We try different settings of attack steps, ϵ and η, selecting the value based on the quality of output triggers. In our experiments, we use 1000 attack steps with $\epsilon = 10$ and $\eta = 1000$.

Final trigger selection Since our process is not deterministic, we initialize multiple independent noise vectors (256 in our experiments) and perform our updates (1) to obtain many candidate triggers. Then, we re-rank the triggers to balance both target classifier accuracy m_1 (lower is better) and naturalness in terms of the average per-token cross-entropy under GPT-2, m_2 (lower is better) using the score $m_1 + \lambda m_2$. We select $\lambda = 0.05$ to balance the difference in scales of m_1 and m_2.

4 Experiments

We demonstrate our attack on two tasks – *sentiment analysis* and *natural language inference*. We use the method of Wallace et al. (2019) as a baseline[4] and use the same datasets and target classifiers for comparison. For the text generator, we use an ARAE model pre-trained on the 1 Billion Word dataset (Chelba et al., 2014). For both our attack (NUTS) and the baseline, we limit the vocabulary of attack trigger words to the overlap of the classifier and ARAE vocabularies. We generate triggers using the development set of the tasks and report results on test set (results on both sets in Appendix).

Defense metrics We employ three simple defense metrics to measure the naturalness of attacks:
1. **Word frequency:** The average frequency of words in the trigger, computed using empirical estimates from the training set of the target classifier.

[4]The baseline attack uses beam search to enlarge the search space in each step. We also tried the baseline attack with 256 random initializations followed by selecting the final trigger using the same criterion as our attack, but its attack success/naturalness remained unchanged.

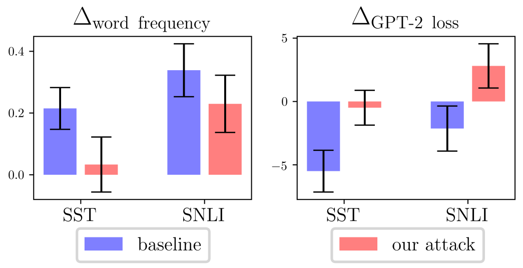

Figure 2: Difference in (a) average word frequency (normalized) and (b) average GPT-2 loss between benign text (x) and different attack triggers (t) (length 8) for SST and SNLI (computed as $stat(x) - stat(t)$). For SNLI, our attacks have lower GPT-2 loss values than even the original text, leading to a positive delta.

Task	*Scribens*		*Chegg Writing*	
	Ours	Baseline	Ours	Baseline
SST	**12.50%**	15.63%	**21.88%**	28.13%
SNLI	**2.08%**	4.17%	**8.33%**	20.83%

Table 1: % of grammatical errors in triggers as per grammar checkers – *Scribens* (scr) and *Chegg* (che).

2. **Language model loss:** The average per-token cross-entropy loss under a pre-trained language model – GPT-2 (Radford et al., 2019).
3. **Automatic grammar checkers:** We calculate the average number of errors in the attack sequences using two online grammar checkers – *Scribens* (scr) and *Chegg Writing* (che).

4.1 Sentiment Analysis

Setup We use a 2-layer LSTM (Hochreiter and Schmidhuber, 1997) followed by a linear layer for sentiment predictions. The model is trained on the binary Stanford Sentiment Treebank (SST) (Socher et al., 2013), using AllenNLP (Gardner et al., 2018). To avoid generating sentiment words in the trigger and directly changing the instance's sentiment, we exclude a list of sentiment words (sen) from the trigger vocabulary, following Wallace et al. (2019).

Results Table 2 (top half) captures the results of both our attack and the baseline (Wallace et al., 2019). Our method is able to reduce the classifier's test accuracy significantly, down to 8.55% in the best attack case. Although less successful, our triggers are much more natural, fluent and readable than the baseline. Figure 2 shows the difference in statistics between benign text and each attack according to the metrics of word frequency and GPT-2 loss. Our generated triggers are much closer in these statistics to the original text inputs than the

Task	Trigger length	Test data	NUTS (our attack) Trigger text	NUTS (our attack) Classifier accuracy	Baseline (Wallace et al., 2019) Trigger text	Baseline (Wallace et al., 2019) Classifier accuracy
SST	No trigger	+	-	89.00%	-	89.00%
		-	-	82.57%	-	82.57%
	8	+	*the accident forced the empty windows shut down*	26.95%	*collapses soggy timeout energy energy freshness intellect genitals*	15.51%
		-	*will deliver a deeply affected children from parents*	8.55%	*sunny vitality blessed lifetime lifetime counterparts without pitfalls*	2.85%
SNLI	No trigger	+	-	89.76%	-	89.76%
		0	-	86.52%	-	86.52%
		-	-	79.83%	-	79.83%
	8	+	*some black women taking the photo last month*	0.00%	*mall destruction alien whatsoever shark pasture picnic no*	0.00%
		0	*the man drowned in hospital and died in*	3.26%	*cats rounds murder pandas in alien spacecraft mars*	0.00%
		-	*they are helping for training achievement for a*	26.78%	*human humans initiate accomplishment energies near objects near*	23.02%

Table 2: Attack results on SST and SNLI. Compared to the baseline, our attacks are slightly less successful at reducing test accuracy but generate more natural triggers. For SST, "+"=positive, "-"=negative sentiment. For SNLI, "+"=entailment , "0"=neutral, and "-"=contradiction. Lower numbers are better. 'No trigger'=classifier accuracy without any attack. Additional attack examples with varying trigger lengths are provided in Appendix.

Condition	Ours	Baseline	Not Sure
Trigger-only	**77.78**	10.93	11.29
Trigger+Benign	**61.16**	21.69	17.15

Text	Natural	Unnatural	Not Sure
Our attack	44.27	50.49	5.24
Baseline attack	22.84	72.00	5.16
Natural text	83.11	14.40	2.49

Table 3: **Human judgement results:** all numbers in %, columns represent the choices provided to human raters. **(Left)** Our attacks are judged more natural than baseline attacks (both on their own and when concatenated with benign input text). Significance tests return $p < 1.7 \times 10^{-130}$ and $p < 4.9 \times 10^{-45}$ for the two rows, respectively. **(Right)** Individual assessments show that our attack is more natural than the baseline but less than benign text on its own (as expected). Significance between natural ratings for our model and baseline has $p < 1.4 \times 10^{-18}$.

baseline. Further, as shown in Table 1, two grammar checkers (scr; che) report 12.50% and 21.88% errors per word on our attack triggers, compared to 15.63% and 28.13% for the baseline.

4.2 Natural Language Inference

Setup We use the SNLI dataset (Bowman et al., 2015) and the Enhanced Sequential Inference Model (ESIM) (Chen et al., 2017) with GloVe embeddings (Pennington et al., 2014) as the classifier. We attack the classifier by adding a trigger to the front of the hypothesis.

Results From Table 2, we see that both our attack and the baseline decrease the accuracy to almost 0% on entailment and neutral examples. On contradiction examples, our attack brings the accuracy down to 26.78% while the baseline decreases it to 23.02%. Although less successful, our attacks are much more natural than the baseline. In Figure 2, our attacks are closer to the word frequency of benign inputs and even achieve a lower GPT-2 loss than the benign text. In Table 1, two grammar checkers (scr; che) also report lower errors on our attacks compared to the baseline.

4.3 Human-Subject Study

To further validate that our attacks are more natural than baseline, we perform a human-subject study on Amazon Mechanical Turk. We collect ratings by: (1) providing a pair of our trigger vs baseline trigger (with and without benign text) and asking the worker to select the more natural one; (2) providing a piece of text (our attack text/baseline attack text/benign input) and asking the human to determine whether it is naturally generated or not. Both conditions allow the human to choose a "Not sure" option. We generated attack triggers with lengths of 3, 5, and 8 (see Appendix for details) and created 450 comparison pairs for (1) and 675 pieces of text (225 for each type) for (2). For each instance, we collect 5 different human judgements and report average scores.

From Table 3 (left), we observe that 77.78% of

Test Class	*Model Architecture*		*Dataset*	
	LSTM ⇓ BERT	BERT ⇓ LSTM	SST ⇓ IMDB	IMDB ⇓ SST
positive	13.91%	41.26%	28.85%	33.67%
negative	51.19%	25.33%	18.13%	30.05%

Table 4: **Attack transferability results:** We report the accuracy drop for our transfer attacks *source-model* ⇒ *target-model*, where we generate natural attack triggers from *source-model* and test their effectiveness on *target-model*. For transferability across model architecture, we use SST as the dataset; for transferability across dataset, we use LSTM as the model architecture.

workers find our attack trigger to be more natural than the baseline while 61.16% judge our attack to be more natural even when concatenated with benign text. The other table shows 44.27% human subjects think our attack inputs are naturally generated. Although it is lower than the 83.11% for real natural inputs, it is still significantly higher than the 22.84% of baseline attack inputs, which shows that our attacks are more natural and harder to detect than the baseline for humans.

4.4 Attack Transferability

Similar to Wallace et al. (2019), we also evaluate the attack transferability of our universal adversarial attacks to different models and datasets. A transferable attack further decreases the assumptions being made: for instance, the adversary may not need white-box access to a target model and instead generate attack triggers using its own model to attack the target model.

We first evaluate transferability of our attack across different model architectures. Besides the LSTM classifier in Section 4.1, we also train a BERT-based classifier on the SST dataset with 92.86% and 91.15% test accuracy on positive and negative data. From Table 4, we can see that the transferred attacks, generated for the LSTM model, lead to 14% $\sim$ 51% accuracy drop on the target BERT model.

We also evaluate attack transferability across different datasets. In addition to the SST dataset in Section 4.1, we train a different LSTM classifier with the same model architecture on the IMDB sentiment analysis dataset, which gets 89.75% and 89.85% test accuracy on positive and negative data. Our attacks transfer in this case also, leading to accuracy drops of 18% $\sim$ 34% on the target model (Table 4).

5 Conclusion

We developed universal adversarial attacks with natural triggers for text classification and experimentally demonstrated that our model can generate attack triggers that are both successful and appear natural to humans. Our main goals are to demonstrate that adversarial attacks can be made harder to detect than previously thought and to enable the development of appropriate defenses. Future work can explore better ways to optimally balance attack success and trigger quality, while also investigating ways to detect and defend against them.

Ethical considerations

The techniques developed in this paper have potential for misuse in terms of attacking existing NLP systems with triggers that are hard to identify and/or remove even for humans. However, our intention is not to harm but instead to publicly release such attacks so that better defenses can be developed in the future. This is similar to how hackers expose bugs/vulnerabilities in software publicly. Particularly, we have demonstrated that adversarial attacks can be harder to detect than previously thought (Wallace et al., 2019) and therefore can present a serious threat to current NLP systems. This indicates our work has a long-term benefit to the community.

Further, while conducting our research, we used the ACM Ethical Code as a guide to minimize harm. Our attacks are not against real-world machine learning systems.

Acknowledgements

We are grateful to the anonymous reviewers at NAACL for valuable feedback. We thank Austin Wang and Michael Hu for suggestions on using Amazon Mechanical Turk.

References

Chegg Writing: Check Your Paper for Free. `https://writing.chegg.com/`.

Opinion mining, sentiment analysis, and opinion spam detection. `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html`.

Scribens: Free Online Grammar Checker. `https://www.scribens.com/`.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang.

2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Pepa Atanasova, Dustin Wright, and Isabelle Augenstein. 2020. Generating label cohesive and well-formed adversarial claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3168–3177.

Melika Behjati, Seyed-Mohsen Moosavi-Dezfooli, Mahdieh Soleymani Baghshah, and Pascal Frossard. 2019. Universal adversarial attacks on text classifiers. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7345–7349. IEEE.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (S&P)*, pages 39–57.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *ACL (1)*, pages 1657–1668. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations (ICLR)*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4422–4431.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing*.

Congzheng Song, Alexander M Rush, and Vitaly Shmatikov. 2020. Adversarial semantic collisions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4198–4210.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Empirical Methods in Natural Language Processing*.

Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*.

Jake Zhao, Yoon Kim, Kelly Zhang, Alexander M Rush, and Yann LeCun. 2018a. Adversarially regularized autoencoders. In *International Conference on Machine Learning (ICML)*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018b. Generating natural adversarial examples. In *International Conference on Learning Representations (ICLR)*.

A Experimental Details

Hyperparameter search For our gradient-based attack approach (Equation (1) in the main paper), there are three hyperparameters: the l_2 norm budget ϵ of the adversarial perturbation, the number of attack steps T, and the step size η in each attack step. Among them, ϵ is super critical for our attacks. A too small ϵ limits the search space over the ARAE (Zhao et al., 2018a) noise input, thus leads to a low attack success. A too large ϵ changes the noise input significantly, thus leads to unnatural trigger generations. In our experiments, we use grid search to manually try different settings of these hyperparameter values: ϵ is selected from $\{2, 5, 10, 20, 50\}$; T is selected from $\{500, 1000, 2000, 5000\}$; and η is selected from $\{10, 100, 1000, 10000\}$. Based on the attack success and the naturalness of generated triggers, we finally set $\epsilon = 10$, $T = 1000$, and $\eta = 1000$.

Dataset and attack details We perform all the attack experiments on a single NVIDIA Tesla P100 GPU. For the sentiment analysis task, we use the binary Stanford Sentiment Treebank (SST) (Socher et al., 2013), which has $6,920$ examples in the training set, 872 examples in the development set, and $1,821$ examples in the test set. The SST classifier uses a two-layer LSTM (Hochreiter and Schmidhuber, 1997) followed by a linear layer for sentiment prediction, with 8.7 million parameters in total. When attacking the SST classifier, it takes around 2 minutes to generate the final trigger.

For the natural language inference task, we use the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015), which has $549,367$ examples in the training set, $9,842$ examples in the development set, and $9,824$ examples in the test set. The SNLI classifier is a pretrained Enhanced Sequential Inference Model (Chen et al., 2017) provided by AllenNLP (Gardner et al., 2018). It has 14.5 million parameters in total. When attacking the SNLI classifier, it takes around 27 minutes to generate the final trigger.

B Additional Experimental Results

B.1 Attack Results with Different Trigger Lengths

Table 5 provides examples of attacks with varying lengths, along with their corresponding classifier accuracies (lower numbers indicate more successful attacks).

B.2 Attack Results on the Development Set and the Test Set

In our experiments, the attack trigger is first generated by increasing the target classifier's loss on the development set, and then applied on the test set to measure its success. Here, we present both the development accuracy and the test accuracy under the same attack triggers in Table 6. We can see that although generated by only attacking the development set, the trigger also works well on the test set: it causes similar accuracy drop on both the development set and the test set.

B.3 Naïve Attacks with Random Triggers

In this section, we check how difficult it is to attack a certain task by implementing two naïve attacks without gradient information. In the first attack method ("Random ARAE"), we randomly collect the candidate triggers generated by the ARAE model (Zhao et al., 2018a), compute the classifier accuracy for each trigger, and finally select the attack trigger as the one with lowest classifier accuracy. We can consider this attack as a simplified version of our attack (NUTS) by removing the gradient information. The second attack method ("Random outputs") is similar as the first one, except that we do not enforce the naturalness of the triggers: we select the attack trigger with the lowest classifier accuracy from many random word sequences. We can also consider this attack as a much simplified version of the baseline attack (Wallace et al., 2019). For both naïve attacks, following our gradient-based attack, we select the final trigger from 256 candidates triggers for a fair comparison.

Table 7 shows all the attack results. First, we observe that these two naïve attacks ("Random ARAE" and "Random outputs") are quite successful in attacking entailment and neutral examples in the SNLI task: they successfully decrease the classifier accuracy to 0% and 3.45%. This indicates that those examples are quite easy to be attacked. Second, for both positive and negative examples in the SST task and the contradiction examples in the SNLI task, the success of these two naïve attacks is quite limited. We also observe a significant improvement on the attack success with these two gradient-based attacks correspondingly.

			NUTS (our attack)		**Baseline** (Wallace et al., 2019)	
Task	**Trigger length**	**Test data**	**Trigger**	**Classifier accuracy**	**Trigger**	**Classifier accuracy**
SST	No trigger	+	-	89.00%	-	89.00%
		-	-	82.57%	-	82.57%
	3	+	*but neither the*	43.01%	*drown soggy timeout*	18.92%
		-	*combined energy efficiency*	19.96%	*vividly riveting soar*	9.10%
	5	+	*a flat explosion empty over*	25.85%	*drown soggy mixes soggy timeout*	10.67%
		-	*they can deeply restore our*	17.11%	*captures stamina lifetime without prevents*	5.26%
	8	+	*the accident forced the empty windows shut down*	26.95%	*collapses soggy timeout energy energy freshness intellect genitals*	15.51%
		-	*will deliver a deeply affected children from parents*	8.55%	*sunny vitality blessed lifetime lifetime counterparts without pitfalls*	2.85%
SNLI	No trigger	+	-	89.76%	-	89.76%
		0	-	86.52%	-	86.52%
		-	-	79.83%	-	79.83%
	3	+	*he was jailed*	0.06%	*alien spacecraft naked*	0.00%
		0	*there is no*	2.52%	*spaceship cats zombies*	0.06%
		-	*he could leave*	54.56%	*humans possesses energies*	47.20%
	5	+	*a man stabbed his son*	0.03%	*alien spacecraft nothing eat no*	0.00%
		0	*there is no one or*	2.27%	*cats running indoors destroy no*	0.00%
		-	*he likes to inspire creativity*	40.07%	*mammals tall beings interact near*	13.44%
	8	+	*some black women taking the photo last month*	0.00%	*mall destruction alien whatsoever shark pasture picnic no*	0.00%
		0	*the man drowned in hospital and died in*	3.26%	*cats rounds murder pandas in alien spacecraft mars*	0.00%
		-	*they are helping for training achievement for a*	26.78%	*human humans initiate accomplishment energies near objects near*	23.02%

Table 5: Attack results on SST and SNLI: Compared to the baseline (Wallace et al., 2019), our attacks are slightly less successful at reducing test accuracy but generate more natural triggers. For SST, "+"=positive, "-"=negative sentiment. For SNLI, "+"=entailment , "0"=neutral, and "-"=contradiction. Lower numbers are better. 'No trigger'=classifier accuracy without any attack.

B.4 Attack Results without GPT-2 Based Reranking

The GPT-2 based reranking is used to balance attack success and trigger naturalness. Without GPT-2 based reranking, the selected trigger will have a slightly higher attack success, however with significantly larger GPT-2 loss. For SST, without reranking, our attack triggers decrease accuracy to 26.84% and 7.68% on positive and negative data, but GPT-2 losses increase from 6.85 (or 6.65) to 8.80 (or 8.88) for positive (or negative) data.

B.5 Variance over Candidate Triggers

For our attacks against negative SST data with trigger length of 8, among all 256 candidate triggers, the average classifier accuracy after attack is 0.23 with a standard deviation of 0.10; and the average GPT-2 loss is 7.93 with a standard deviation of 0.85. There is no inherent tradeoff between naturalness and attack success: some triggers have both low classifier accuracy and low GPT-2 loss, and the pearson correlation is -0.08.

C Human-Subject Study Details

We perform the human-subject study on Amazon Mechanical Turk. Crowdworkers were required to have a 98% HIT acceptance rate and a minimum of 5000 HITs. Workers were asked to spend a maximum of 5 minutes on each assignment (i.e., comparing the naturalness of a pair of our trigger vs baseline trigger, or evaluating the naturalness of a piece of text), and paid $0.01 for each assignment.

Task	Trigger length	Data	NUTS (our attack) Accuracy (dev set)	NUTS (our attack) Accuracy (test set)	Baseline Accuracy (dev set)	Baseline Accuracy (test set)
SST	No trigger	+	88.29%	89.00%	88.29%	89.00%
		-	82.94%	82.57%	82.94%	82.57%
	3	+	40.54%	43.01%	20.27%	18.92%
		-	21.26%	19.96%	10.51%	9.10%
	5	+	26.35%	25.85%	12.39%	10.67%
		-	18.46%	17.11%	6.31%	5.26%
	8	+	27.25%	26.95%	17.79%	15.51%
		-	10.05%	8.55%	1.87%	2.85%
SNLI	No trigger	+	90.96%	89.76%	90.96%	89.76%
		0	88.07%	86.52%	88.07%	86.52%
		-	79.53%	79.83%	79.53%	79.83%
	3	+	0.03%	0.06%	0.00%	0.00%
		0	2.53%	2.52%	0.00%	0.06%
		-	54.58%	54.56%	46.55%	47.20%
	5	+	0.00%	0.03%	0.00%	0.00%
		0	1.82%	2.27%	0.00%	0.00%
		-	39.48%	40.07%	13.24%	13.44%
	8	+	0.00%	0.00%	0.00%	0.00%
		0	3.74%	3.36%	0.00%	0.00%
		-	25.90%	26.78%	22.76%	23.02%

Table 6: Universal attack results on both the development (dev) set and the test set for the Stanford Sentiment Treebank (SST) classifier and the Stanford Natural Language Inference (SNLI) classifier. For SST, "+"=positive, "-"=negative sentiment. For SNLI, "+"=entailment , "0"=neutral, and "-"=contradiction. We first generate the attack trigger by increasing the classifier's loss on the dev set, and then apply the same trigger on the test set. 'No trigger' refers to classifier accuracy without any attack. We can observe that the same triggers achieve similar attack success in both the development set and the test set.

Attack method	**Trigger length**	**SST**		**SNLI**		
		Positive	**Negative**	**Entail-ment**	**Neutral**	**Contrad-iction**
No attack	-	89.00%	82.57%	89.76%	86.52%	79.83%
NUTS (Our attack)	3	43.01%	19.96%	0.06%	2.52%	54.56%
	5	25.85%	17.11%	0.03%	2.27%	40.07%
	8	26.95%	8.55%	0.00%	3.26%	26.78%
Random ARAE	3	54.46%	66.78%	0.09%	11.59%	58.02%
	5	50.28%	43.75%	0.00%	13.36%	55.51%
	8	43.23%	39.69%	0.03%	8.01%	42.79%
Baseline attack	3	18.92%	9.10%	0.00%	0.06%	47.20%
	5	10.67%	5.26%	0.00%	0.00%	13.44%
	8	15.51%	2.85%	0.00%	0.00%	23.02%
Random outputs	3	49.17%	32.79%	0.36%	17.43%	61.48%
	5	47.19%	23.90%	0.00%	3.45%	53.35%
	8	41.58%	20.07%	0.00%	7.80%	50.82%

Table 7: Universal attack results on both the Stanford Sentiment Treebank (SST) classifier and the Stanford Natural Language Inference (SNLI) classifier. Besides gradient-based attacks including our attack (NUTS) and the baseline attack (Wallace et al., 2019), we further implement two naïve attacks without gradient-guided search: "Random ARAE" means we select the best attack trigger from random natural ARAE outputs; "Random outputs" represents we select the best attack trigger from random unnatural word sequences.