Fooling GPT with adversarial in-context examples for text classification

Sudhanshu Ranjan
University of California San Diego, Lindy.ai [*]
sranjan@ucsd.edu

Chung En Sun
University of California San Diego
cesun@ucsd.edu

Linbo Liu
University of California San Diego
linbo@ucsd.edu

Tsui-Wei Weng
University of California San Diego
lweng@ucsd.edu

Abstract

Deep learning-based methods helped solve NLP tasks more efficiently than traditional methods and adversarial attacks for these methods have been extensively explored. However, Large Language Models (LLMs) have set up a new paradigm of few-shot prompting, opening up the possibility for novel attacks. In this study, we show that LLMs can be vulnerable to adversarial prompts. We develop the first method to attack the few-shot examples in the text classification setup. We can degrade the model performance significantly during the test time by only slightly perturbing the examples based on optimization. Our method achieves a performance degradation of up to 50%. The perturbed examples are semantically similar to the original examples as shown by the sentence similarity score.

1 Introduction

Deep Neural Networks (DNNs) achieved good end-to-end performance on Natural Language Processing (NLP) tasks without handcrafted features. Recently, the emergence of transformer architectures made language models even more powerful. BERT [6] is able to solve a variety of downstream tasks through finetuning on the corresponding dataset with only a little cost. GPT [16] established a new prompting paradigm that enables users to provide instructions and few-shot examples as input. With only human instructions and a few labeled examples in the context, the GPT model is able to perform the corresponding task and gives high-quality answers. Furthermore, instruction-tuned LLMs [19, 5, 17] can solve NLP tasks including language generation even in zero-shot setting.

Adversarial robustness of the previous DNN-based methods has been extensively explored [8, 9, 3, 14]. However, the new paradigm of prompting makes it possible to perform novel attacks. Recent studies have shown that the GPT model might be vulnerable to handcrafted prompts. [15] showed that GPT can be easily misled by human-made instructions telling the model to perform a totally unrelated task. Their simple methodology demonstrated that users can attack GPT without using complicated algorithms. Although powerful, adversarial instructions can be easily detected by humans.

We instead propose a method to generate adversarial perturbations for the in-context examples without human intervention. A large number of applications involve situations where the victim would be solving the text classification task (for example topic prediction) using the same base model and in-context samples across days. The attacker only needs to run the attack once and perturb the in-context examples. This setting is similar to backdoor attacks [3, 14] where the attacker can poison the training data of the victim. In this case, the attacker is perturbing the in-context examples.

[*]Work done while at UCSD.

Furthermore, we believe our setting is more reasonable since few-shot examples are often long and not provided by humans. Our adversarial examples barely change the content and hence are hard to be detected by humans. However, they are able to fool the text classification result. To the best of our knowledge, we are the first to show that it is possible to degrade the performance of LLMs significantly through word substitutions in the few-shot examples. Our method has the following strengths:

- The adversarial examples are hard to detect and preserve the original semantic meanings.
- Our attack can reduce the model performance by up to 50%.
- Our pipeline is fully automatic.

We formally introduce our method in the section 3.

2 Related works

Adversarial perturbations in NLP TextFooler [8] proposed a method using synonyms and other heuristics for generating imperceptible adversarial texts. More specifically, given a sample during the inference time, they generate an adversary that stays similar to the sample but fools the system. [9, 7] propose using BERT-based substitutions, leading to better substitution strategy and more successful attacks.

Backdoor attacks in NLP Inspired by backdoor attacks in computer vision, [3] proposed backdoor attacks in NLP. The attacker poisons the training data of the victim by adding secret triggers. During inference time, texts with these triggers would fool the model. [14] propose a backdoor attack based on manipulating the style of the text. The attack operates by rephrasing some of the training samples using an alternate stylistic form, such as Shakespearean or poetic language. Test time samples that use these styles are able to fool the model. While the defenses for trigger-based backdoor attacks have been explored [11], defenses for style-based backdoor attacks are still an open problem.

Robustness of LLMs LLMs generate objectionable content when used "out-of-the-box" and researchers have used alignment [13, 1] to prevent the same. [23] showed that it is possible to find triggers using optimization which when appended to user query can lead to LLMs generating objectionable content. [18] also show that it's possible to attack GPT models to make them generate toxic and biased outputs, and leak private information in training data. [4] study the robustness of GPT models during test time inference. [20, 12] quantify the uncertainty in prediction and language generation tasks respectively.

3 Methodology

Problem Formulation We focus on text classification using LLMs in the few-shot prompting paradigm. The victim uses an LLM M and has a prompt c which consists of an instruction I and few-shot example sentence-label pairs $\mathcal{D} = \{(s_i, y_i)\}$. To make an inference for a test sentence t, it is appended to the prompt c, and the concatenated string is passed as input to M to predict the class for t. Let $M(x, c)$ be the output of the LLM for the sample x when prompt c is used. The attacker knows the model M to be used by the victim, can modify the samples in c, and has access to additional labeled data D_{dev}. The attacker's aim is to modify c to get c' such that the adversarial prompt c' is semantically similar to c and can reduce the model performance significantly during the inference. We don't modify the instructions or the labels of the examples, since humans write instructions and the changes can be detected while changes to the labels would make the example incorrect. Thus, only s_i is modified by our algorithm.

Our methodology consists of three main steps. It relies on relevance score, a metric used to compare two versions of the prompt and decide which would decrease the model performance more. We define relevance score formally in section 3.1. The relevance score of a word and its substitutes is defined using this concept later. The three steps are:

1. Finding the substitutes for the words in the in-context examples using Masked Language Modelling (MLM). These substitutes are used later to get the adversarial examples.

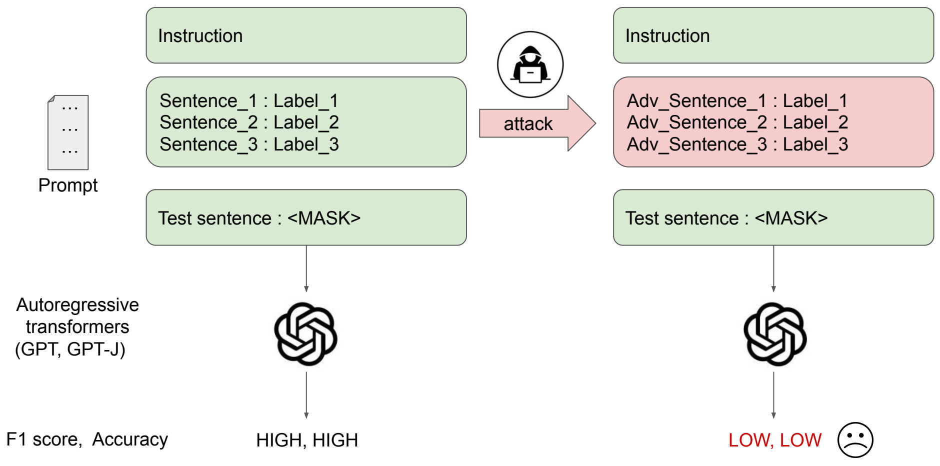

Figure 1: Overview of the proposed attack. The prompt consists of instructions, few-shot example sentences, few-shot example labels, and a test sentence. An attacker only adversely alters the few-shot example sentences into **adversarial sentences** by substituting a fixed number of words selected by our algorithm while leaving the other components unattacked. Empirical experiments show that the **adversarial sentences** can significantly decrease the model performance in terms of F1 score and accuracy. Furthermore, **adversarial sentences** are semantically similar to the original sentences.

2. Getting the relevance score for each word that can be modified. Using the relevance score for the words, we can determine the order in which we should modify the words.
3. Use the relevance score to choose the substitute leading to a maximum drop in the model performance during test time inference.

3.1 Relevance score

We define the relevance score of a prompt as the average decrease in the probability of the correct class across the samples in the development split D_{dev}. The objective of our method is to decrease the classifier's performance during the inference time. Thus, the relevance score of the current prompt should indicate how much the classifier's performance would degrade compared to the original prompt. Given two prompts c_1 and c_2, if the relevance score of c_1 is higher, we should expect the model to perform lower when c_1 is used as the prompt compared to the prompt c_2.

More formally, let C be the set of all possible prompts in length ℓ and REL : $C \to [0, 1]$. The relevance score of the adversarial prompt is defined as follows:

$$\text{REL}(c') = \frac{\Sigma_{(x,y)\in D_{\text{dev}}}(p_M(y|x,c) - p_M(y|x,c_{\text{adv}}))}{|D_{\text{dev}}|}, \tag{1}$$

where c is the original prompt, c_{adv} is the adversarial prompt and $p_M(y|x,c)$ is the probability of predicting class y when inputs to LLM M are the sample x and prompt c.

3.2 Masked Language Modelling (MLM) based substitutions

This step aims to find substitutes for the words such that the new sentence is semantically similar to the original sentence. We use the bidirectional language model BERT [6] along with MLM objective to get the top k possible substitutes for each word in the few-shot example sentences of the prompt. Compared to using the synonyms of the words, this method finds more substitutes while keeping the semantics the same and the grammar coherent [7, 9]. We also use Universal Sentence Encoder (USE) [2] to get the similarity scores between new and previous sentences and do not consider substitutes where the similarity score is less than the threshold τ. This helps filter out the substitutes that change

Algorithm 1: Adversarial Attack on the In-Context Examples

```
Input: LLM $M$, prompt $c$, development split $D_{\text{dev}}$, perturbation percent $p$, substitute count $k$, semantic
  threshold $\tau$.
Output: Adversarial prompt $c_{\text{adv}}$.
// Substitute Finding and Relevance Score Computation
RelevanceList $\leftarrow$ [ ]
for i $= 0$ to $len(c)$ do
    $S_i \leftarrow$ GetCandidates$(c, w_i, k)$ // Get candidates using BERT.
    $S_i \leftarrow$ FilterSubstitutes$(S_i, c, \tau)$ // Filter set of substitutes with USE.
    RelevanceList$[i] \leftarrow$ (REL$(c_{\text{adv, i}})$, $i$) // Compute relevance score.
// Sort in the order of decreasing relevance score
RelevanceListSorted $\leftarrow$ ReverseSort(RelevanceList)
// Finding the Adversarial Prompt
$c_{\text{adv}} \leftarrow c$
countModified $\leftarrow 0$ // Maintain count of number of words modified
for i $= 0$ to $len(RelevanceListSorted)$ do
    $j \leftarrow$ RelevanceListSorted$[i][1]$
    $m \leftarrow \arg\max_t$ REL$(c_{\text{adv,j,t}})$
    $c_{\text{adv}} \leftarrow c_{\text{adv,j,m}}$
    countModified $\leftarrow$ countModified $+ 1$
    if $countModified \geq len(c) * p\%$ then
        Return $c_{\text{adv}}$
```

the semantics of the sentence. Formally, let S_i denote the set of substitutes of the word w_i. Then $S_i = \{w_{i,1}, w_{i,2}, \ldots, w_{i,k_i}\}$ where $w_{i,1}, w_{i,2}, \ldots, w_{i,k_i}$ denotes the k_i possible substitutes of word w_i. k_i could be different for each i. We use S_i in step 3 to find the best substitute.

3.3 Relevance score-based ordering

The relevance score of a word is obtained by substituting the word in the original prompt with the <MASK> token and finding the relevance score of the newly created prompt. We find relevance scores for all the words in the prompt. Similar to [7], [9] and [8], we then sort the words based on decreasing order of relevance score. Intuitively, substituting words having a higher relevance score would result in prompts that decrease the model's performance to a greater extent.

Formally, let w_i be the i-th word in the prompt c. Let $c_{\text{adv, i}}$ be the new prompt where all words are the same as c but i-th word had been substituted by <MASK>. Let $j_1, j_2, ..., j_l$ be the indices of words in decreasing order of relevance score i.e. $\text{REL}(c_{\text{adv, j}_1}) \geq \text{REL}(c_{\text{adv, j}_2}) \geq ... \geq \text{REL}(c_{\text{adv, j}_l})$.

3.4 Modifying the in-context examples

We make substitutions based on the relevance score. Assume we currently have an adversarial prompt. For a given word, we have a set of substitutes found in step 1 (section 3.2). We create a new prompt for all possible substitutes by replacing the word with the substitute in the current adversarial prompt. We then calculate the relevance score for all newly constructed prompts and choose the substitute whose corresponding prompt has the highest relevance score. Once the new adversarial prompt is obtained by substituting the given word with the appropriate substitute, we use this prompt as the adversarial prompt to find the adversarial substitute for the next word.

More formally, let c_{adv} be the current adversarial prompt, and let w_j be the word with the highest relevance score that hasn't been substituted yet. Then, $w_{j,1}, w_{j,2}, \ldots, w_{j,k_j}$ are the possible substitutes. Now construct the new prompts $c_{\text{adv,j,1}},\ c_{\text{adv,j,2}},\ \ldots,\ c_{\text{adv,j,k}_\text{j}}$ by substituting the word w_j in the original c_{adv} prompt with $w_{j,1}, w_{j,2}, \ldots, w_{j,k_j}$ respectively. Let m be the index of the substitute whose corresponding prompt has the highest adversarial score, i.e., $m = \text{argmax}_t\ \text{REL}(c_{\text{adv,j,t}})$. We consider $c_{\text{adv,j,m}}$ as the new adversarial prompt c_{adv}. Then, we consider the next word, which has not been substituted yet, and repeat the process.

The algorithm of our attack is in Algorithm 1 .

Table 1: Results of attack on GPT2-XL across different datasets.

	Agnews		Dbpedia		TREC		Yelp	
$\delta\%$	F1 (Pct $\downarrow$)	Acc (Pct $\downarrow$)	F1 (Pct $\downarrow$)	Acc (Pct $\downarrow$)	F1 (Pct $\downarrow$)	Acc (Pct $\downarrow$)	F1 (Pct $\downarrow$)	Acc (Pct $\downarrow$)
0	36.4 (0.0)	46.2 (0.0)	46.0 (0.0)	50.7 (0.0)	38.5 (0.0)	46.1(0.0)	67.7 (0.0)	69.5 (0.0)
1	33.8 (6.7)	46.1 (0.3)	35.5 (25.6)	42.0 (8.0)	35.8 (7.4)	44.1(4.2)	63.1 (6.8)	65.8 (5.2)
5	23.4 (37.9)	35.2 (24.3)	24.6 (46.2)	32.2 (34.9)	29.4 (25.7)	38.4(17.8)	52.1 (24.1)	58.6 (15.8)
10	17.9 (52.3)	29.6 (35.9)	12.5 (70.5)	22.6 (52.0)	27.7 (29.4)	36.3(21.9)	46.4 (32.0)	54.8 (20.8)

Table 2: Results of attack on GPT-Neo-2.7B and Llama-2-7B on Dbpedia dataset.

	GPT-Neo-2.7B		Llama-2-7B	
$\delta\%$	F1 (Pct $\downarrow$)	Acc (Pct $\downarrow$)	F1 (Pct $\downarrow$)	Acc (Pct $\downarrow$)
0	53.9 (0.0)	58.8 (0.0)	39.1 (0.0)	43.7 (0.0)
1	47.9 (11.2)	54.5 (7.3)	37.9 (3.0)	42.4 (2.8)
5	41.1 (26.9)	49.3 (17.6)	37.1 (5.3)	41.7 (4.6)
10	35.3 (37.2)	44.4 (25.5)	35.8 (7.5)	40.8 (5.9)

Table 3: Average semantic similarity of the perturbations with original sentences measured using USE.

$\delta\%$	Agnews	Dbpedia	TREC	Yelp
0	1.00	1.00	1.00	1.00
1	0.96	0.95	0.95	0.95
5	0.89	0.89	0.93	0.88
10	0.83	0.84	0.91	0.82

4 Experiments

We describe the details relevant to the experiments in this section.

4.1 Experimental settings

We use four datasets for testing the performance of our attack: Agnews [21], Dbpedia [21], Yelp [21] and TREC [10]. We conducted the experiments with three different seeds and reported the average of the results. We use GPT2-XL in our experiments. We calibrate as proposed in [22] before getting the final label. The method helps in mitigating the class bias for LLMs and helps in getting better initial performance. We report accuracy, weighted F1 score, and percent decrease in the model's performance. Perturbation percent reflects the number of words that have been perturbed among the initial set of words that could have been perturbed. We report the values at three perturbation percentages: 1%, 5%, and 10%.

4.2 Attack results

The results are reported in table 1. We observe that our model can make the performance deteriorate across all the datasets. We observe that the maximum decrease comes when the attack is done on datasets related to topic classification. Across the datasets, the attack performs best on the Dbpedia dataset, and the F1 score decreases up to 70%. We believe our results are significant because even with 1% perturbation percent, we observe at least a 4% performance decrease across datasets. We also experiment with bigger models GPT-Neo-2.7B and Llama-2-7B on the Dbpedia dataset, and the results are present in table 2. Even though the models are more robust, we observe that our attack can decrease the performance of the models by more than 25% and 7% respectively.

4.3 Qualitative analysis

We show a sample perturbation in Figure 2. Since the words are substituted by MLM objective, the substituted words aren't necessarily the synonyms of the original word but still keep the sentence coherent and semantically similar. For example, the word *Malaysian* is replaced by *Asian*, and *Safety* is replaced by *portable*. Given the semantic similarity of the original and the perturbed sentences, it logically follows that the labels for the perturbed sentences should be the same as those of the original sentences. We observe that for Dbpedia and Agnews datasets, our method chooses to substitute the proper nouns more frequently than other parts of the sentence. Quantitatively, the semantic similarity between two sentences can be measured using USE. We report the average semantic similarity between modified and original sentences of the few-shot examples in table 3. We observe that the average semantic similarity across the datasets stays above 0.8 in all the cases.

Article: Ex-Enron Official Details Merrill Lynch Deal. HOUSTON, Oct. 7 – The government prepared Thursday to rest its fraud and conspiracy case against four former Merrill Lynch Co. executives and two former mid-level Enron Corp. executives, following testimony from a key Enron executive that a barge forced sale was a sham prank designed to inflate profits. Answer: Business

Article: PowerTech Inc. is a small business located in Collierville Tennessee. It is a privately owned and operated corporation. Powertech is the exclusive licensee (manufacturer) for Smith Wesson Flashlights. For one year (2004–05) the company produced Coleman military Safety portable Products. Answer: Company

Question: Shea lewis and Gould closed their Los Angeles office for what reason? Answer Type: Description

Review : I have been to this place a few times and enjoyed the flavors. However I was VERY DISAPPOINTED for their lack of "respect" to what a customer orders. I order my usual the MEE GORENG with TOFU and as I combed through and had a couple of bites I found a strip of chicken. I lost my appetite and was gravely disappointed by their lack of detail to a vegetarian dish. This place is black listed for me and I hope to find a better Malaysian asian place in the Queen City. Sentiment: Negative

Figure 2: Example of a perturbed few-shot sample from Agnews, Dbpedia, TREC, and Yelp datasets. Red and blue colors denote words that have been deleted and inserted, respectively.

Semantic similarity of the perturbed sentence with the original sentence is controlled by the parameter τ. The higher the value of the parameter τ, the more the semantic similarity of the perturbed sentence with the original sentence. A higher value of τ implies a smaller number of possible substitutes for any given word and hence a smaller set of perturbations for any given sentence to choose from. Thus, there is a trade-off between the quality of the perturbations and the attack performance.

5 Conclusion and future work

We demonstrate that LLMs can be easily fooled with small perturbations in the few-shot examples, which are semantically similar to the original ones and hard to detect. Our study opens a new issue on the robustness of LLMs against the few-shot adversarial examples. There are two possible directions for future work: 1) improve the efficiency of our attack by using surrogate models that approximate the relevance score for a given prompt and reduce the run time, and 2) enhance the robustness of few-shot prompting-based paradigms based on our proposed attack.

References

[1] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022.

[2] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018.

[3] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. 2021.

[4] Xuanting Chen, Junjie Ye, Can Zu, Nuo Xu, Rui Zheng, Minlong Peng, Jie Zhou, Tao Gui, Qi Zhang, and Xuanjing Huang. How robust is gpt-3.5 to predecessors? a comprehensive study on language understanding tasks, 2023.

[5] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

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

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

[8] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[9] Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.

[10] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[11] Zichao Li, Dheeraj Mekala, Chengyu Dong, and Jingbo Shang. BFClass: A backdoor-free text classification framework. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021.

[12] Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. Generating with confidence: Uncertainty quantification for black-box large language models, 2023.

[13] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[14] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. Hidden trigger backdoor attack on NLP models via linguistic style manipulation. In *31st USENIX Security Symposium (USENIX Security 22)*, 2022.

[15] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *CoRR*, 2022.

[16] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[17] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization, 2022.

[18] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models, 2023.

[19] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Maitreya Patel, Kuntal Kumar Pal, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Shailaja Keyur Sampat, Savan Doshi, Siddhartha Mishra, Sujan Reddy, Sumanta Patro, Tanay Dixit, Xudong Shen, Chitta Baral, Yejin Choi, Noah A. Smith, Hannaneh Hajishirzi, and Daniel Khashabi. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks, 2022.

[20] Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis, 2022.

[21] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657, 2015.

[22] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 18–24 Jul 2021.

[23] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

A Appendix

For our experiments, we use a D_{dev} of size 100 samples and report the final performance on a test split of size 300 samples. We do the experiments with 3 seeds and report the average performance across them. While making the predictions, we use the greedy method to choose the most likely class.