AlphabetSoup: Generating High Confidence Fooling Texts for NLP Models

Daniel Ezer
danielezer@mail.tau.ac.il
315071506

Leah London Arazi
leahl@mail.tau.ac.il
206766644

ABSTRACT

Language Models are integrated into increasingly sensitive systems, and thus the motivation for attacking them is growing. Most attacks use adversarial examples, which involve slight modifications of the original input to alter the model's output. In this project, we describe a different approach, finding "fooling texts", which are random-looking texts that are classified with high confidence.

We propose two attack methodologies: *Random Chaos* and *Patterned Chaos*. The former generates unstructured, random prompts, while the latter produces structured, yet incoherent text. Together, these methodologies include a total of four attacks: *Character Roulette* replaces individual characters or words in both white-box and black-box settings. *Unbounded Drift* employs recent gradient-based optimization methods. *Syntactic Sabotage* generates prompts with some syntactic structure, and *Meaning Masquerade* produces prompts from a vocabulary of semantically related words.

Our results demonstrate that "fooling texts" can be efficiently generated using all of our attacks, even in black-box setting, with partial success. Our more successful attack, Unbounded Drift, achieves close to 100% success rate, and generates nonsensical texts that perplex the strong GPT-2 model more than random text. The Patterned Chaos attacks are also very successful, most of which achieve high accuracy scores with a very limited set of tokens.

1 INTRODUCTION

Machine Learning models are common and useful for many domains, such as image classification and text generation. As these models become available to more users, they are also likely to be attacked.

Test-time attacks on computer vision models are widespread since the work of [10], which shows that it is possible to make models classify images as arbitrary classes using small imperceptible perturbations. Apart from being interesting, these examples provide a window to the inner-workings of the model, and could help design more reliable models.

The work of [20] has shown that it is easy to find *"fooling images"*, which are seemingly random images that are classified with high confidence as some class. While the two approaches are related, they demonstrate different shortcomings of vision models, suggesting that these models probably don't "understand" images as robustly as we hope.

A complementary line of work focuses on attacking NLP models used for classification tasks. Unlike images, in natural language it is not possible to perturb the input in a way that is completely indistinguishable to humans. Hence, attacks in this field attempt to make small and coherent changes that alter the model's behaviour arbitrarily, while being syntactically or semantically similar to the original text. Some work change a few characters [7, 9], others change words while preserving the semantics [14], but all follow the same concept as adversarial examples in computer vision.

Other than classification, "jail-breaking" attacks on generative language models try to make them generate content which should typically be restricted. [32] were able to provoke toxic and dangerous behaviour from state-of-the-art models, by automatically finding a prompt that makes them ignore their restrictions. The generated prompts, unlike adversarial examples, seem unnatural and can easily be spotted by human evaluators.

Our work aims to extend existing attacks in the realm of NLP, by finding *"fooling texts"* which are seemingly random, nonsensical textual inputs that are classified with high confidence. Inspired by [20], we aim to produce such examples that appear as random gibberish, as well as inputs that adhere to a certain structure. We call the former attacks *Random Chaos* and the latter *Patterned Chaos*.

2 PREVIOUS WORK

2.1 Fooling images

Our project is inspired by the work of [20], which generate *"fooling images"* to mislead Deep Neural Networks (DNN) to confidently classify images that are unrecognizable to humans. In order to do so, their work uses evolutionary algorithms (EAs) and gradient ascent. Furthermore, the paper shows that it is possible to fool models by using images with repetitive patterns, that appear unrelated to the input class. Both methods are able to fool DNNs trained on MNIST [4] and ImageNet [3] with very high Success Rates. Their work heavily relies on the fact that images are represented in continuous space, and are easily optimized using standard optimization methods. This is in contrary to text, which is inherently discrete, and thus harder to optimize.

2.2 Large Language Models (LLMs)

Large Language Models (LLMs) have become increasingly useful for performing various tasks, ranging from sentiment analysis and toxicity classification of social media posts to medical advice. These applications often involve sensitive data and have the potential to cause significant harm. Thus, in an attempt to understand their limitations and downfalls, attacking LLMs is becoming more important, with the goal of producing safer and more robust models.

LLMs take tokens, which are the basic units of text, as input. Tokens can be words, sub-words or characters, and are created by tokenizers. Language models are usually pre-trained on a large corpus of data, and some of them are available for public use. Pre-training usually takes one of two approaches: Masked Language Modeling (MLM) or Language Modeling (LM).

BERT [5] is an encoder-only transformer [28] that uses the MLM objective. It takes tokens as input and outputs an embedding, which is a representation of the input sequence. In order to use BERT for classification, a MLP head leverages the information in BERT's embeddings to perform different tasks, achieving high accuracy [27]. RoBERTa [17] and DeBERTa [12] are variants of BERT which even further increase its accuracy on standard NLP benchmarks such as GLUE [29] and SQUAD [24]. Nowadays, most text classification models use BERT-based architectures, as evident in the Hugging Face Model Hub[1].

In this work, we attack different text classification models that use fine-tuned versions of BERT for sentiment analysis and toxicity classification.

2.3 Prompting LLMs

Previous work has found that specific prompts can be used in order to elicit different model behaviors, even surpassing fine-tuning [2]. Building on these findings, In-Context-Learning, which attempts to manually find prompts that help models perform better, has become pervasive [6]. Since manually finding prompts is brittle and time consuming, attempts have been made to automate this method.

However, due to the discrete nature of tokens, using naive optimization methods is highly non-trivial. An alternative may require querying the model using many different prompts, which quickly becomes computationally infeasible. Hence, efficient automation methods such as *soft prompting* and *hard prompting* have emerged.

Soft prompting uses a non-textual continuous embedding in the token embedding space, which can be found by leveraging gradient based optimization methods [16]. While soft prompts show some promise, they are usually hard to interpret as textual prompts and are not transferable to models with different embedding spaces [15]. This also renders these prompts are ineffective for models that only provide API access.

Hard prompting, in contrast, leverages methods for discrete optimization on the input tokens to find textual prompts. Some methods represent input tokens as one-hot vectors and use them to estimate the gradients [7]. AutoPrompt [26] uses this representation to iteratively find the top-k most influential replacements, and then changes the prompt according to the replacement which minimizes the loss. Adding the resulting prompt to the input achieves results that are comparable to task specific fine-tuning.

GCG [32, Algorithm 1] is an improvement to the AutoPrompt algorithm used for automatically jailbreaking generative models by making them predict an initial affirmative response as a response for the given prompt. This approach causes the models to generate toxic and harmful content, and is successful in attacking models such as ChatGPT [21] and Claude [1].

However, since this method requires numerous forward passes for choosing the replacement, it can become costly. PEZ [30] uses a hybrid approach reminiscent of Projected Gradient Descent (PGD). Their algorithm performs gradient steps in continuous space, and projects the resulting embeddings at each iteration to its nearest neighbor in the vocabulary.

2.4 Adversarial Examples in NLP

In the field of NLP, researchers have attempted to replicate similar phenomena as observed in the vision domain by creating minor changes to natural language, while preserving its coherency and semantics.

Some works achieve this goal using black-box attacks. [13] make simple character level modifications to fool Google's toxicity classification API. DeepWordBug [9] identifies the crucial tokens that, when altered, cause the classifier to make an incorrect prediction. They then perform a single iteration on the input text, changing at most a single character per word, starting with the most influential words. This results in text which is similar to the original input, with only a handful of different characters. [25] attack a text to gender classifier by replacing words with synonyms. Likewise, TextFooler [14] misleads sentiment classifiers by replacing the word that most strongly affects the original label with a similar word, while preserving syntactic coherency.

Other work use white-box attacks which leverage gradients to choose the most influential changes. HotFlip [7] can be seen as a white-box version of DeepWordBug [9]. They use an efficient attack on character-level language models, by representing words as a combination of one-hot vectors.

Many of the aforementioned attacks are implemented with TextAttack [19], a python framework that helps researchers attack NLP models. The attacks try to increase an *objective function* by perturbing texts using a *transformation* with respect to a set of *constraints*. A *search method* is used to choose options from the generated perturbed texts in each iteration. TextAttack is easy to use and contains implementations for 16 attacks, including DeepWordBug [9] and HotFlip [7]. We rely heavily on TextAttack in this project to implement our attacks.

3 ATTACKS AND TECHNICAL APPROACH

This project involves implementing and testing multiple variations of test-time, targeted text classification attacks in both white-box and black-box settings. The source code can be found at our GitHub repository.

In this section, we introduce the *Random Chaos* and *Patterned Chaos* attack families. The former includes attacks that produce seemingly random text, resembling incomprehensible gibberish. The latter consists of attacks that generate text with certain patterns, either semantic or syntactic.

Before discussing the attacks in detail, we first present our threat model. In the *white-box* setting, we assume the adversary has full access to the model, and can therefore calculate the gradient with respect to the input embeddings.

In the *black-box* setting, the adversary is assumed to have no access to the model parameters or training data, and can only query the model for confidence scores. This threat model is usually more realistic, and therefore interesting to explore.

In both settings, it is assumed that the adversary knows the vocabulary and the tokenizer used by the model.

3.1 Random Chaos Attacks

In this family of attacks, we conduct experiments with two types of input initializations: random initialization, where we sample 20 random words, and initialization with 20 repeated instances of a single word. We arbitrarily chose the word *"aa"*.

[1]https://huggingface.co/models

3.1.1 ***Character Roulette.***

This attack is inspired by DeepWordBug [9] and HotFlip [7]. We make multiple passes over the current text, identify the most impactful words, and modify them using various methods.

In the white-box setting, we leverage the gradient w.r.t the input text in order to make the most influential tokens replacement. At each iteration, we calculate the gradient of the target class classification loss w.r.t to the current text embedding. For every token embedding **a** in the current text embedding, we calculate the gradient in the direction of the swap of **a** with another token embedding **b**. This serves as an estimate for the loss decrease of swapping **a** with **b**, allowing to choose promising candidates. HotFlip uses a similar approach, but is limited to attacking models with character-level or word-level tokenizers, making it unsuitable for most modern setups that use sub-word tokenizers. By calculating the gradient w.r.t token embedding, we extend the attack's applicability to state-of-the-art text classifiers.

In the black-box setting, we follow the method proposed by DeepWordBug to rank the importance of words, and apply perturbations to change the classification output. we are not concerned with preserving the meaning of the input text, so we remove all constraints on the perturbations. We perform multiple passes over the input text, identifying influential words in each pass and modifying each word multiple times. At each pass, we choose to apply the replacement with the highest target class confidence. Furthermore, in an attempt to make our search less strict, we added a swap threshold that allows the attack to perturb the text even if it does not improve the goal function.

We carried out the black-box attack in two ways: one with character-based perturbations and the other with word-based replacements. The character-based approach involved replacing an existing character with a random one, swapping neighboring characters, inserting a random character, or deleting a character. The word-based approach simply replaced a word with a randomly sampled word.

3.1.2 ***Unbounded Drift.***

This attack takes two forms, based on the work of GCG [32] and PEZ [30].

Our PEZ-inspired attack uses the optimization algorithm from PEZ [30, Algorithm 1], leveraging gradient-based techniques in the token embedding space in order to find an input text which results in confident classification. To the best of our knowledge, this is the first adaptation of PEZ to an attack.

The GCG-based attack adapts the GCG attack [32, Algorithm 1] from generative models to text classification models. Instead of targeting an initial affirmative response, we target a specific output class and perform token replacements to find a "fooling text".

These attacks allow the input text to change to change arbitrarily, with the sole objective of increasing prediction confidence. We initially expected this to produce seemingly random tokens. However, the unconstrained approach usually causes the optimization algorithm to find ungrammatical texts that contain words that are clearly correlated with the target class. This interesting finding motivated us to limit the tokens available for replacements during the attack, as detailed in the next section.

3.2 Patterned Chaos Attacks

The following attacks are a constrained version of those presented in Section 3.1.2. Unlike in Section 3.1, this attack family does not use an initial random prompt, since this often led to the inclusion of words outside our constrained set of tokens. Therefore, we initialized the prompt by repeating the arbitrary word "aa".

3.2.1 ***Syntactic Sabotage.***

In this attack, our goal is to generate text that appears nonsensical but still follows some *syntactic patterns*. Initially, we aimed to guide the optimization method towards prompts with low character-level entropy. However, because tokenization is inherently non-differentiable, it is not possible to add a regularization term based on the text itself—only on the embeddings.

As an alternative, we employed a different strategy where the attack is restricted to using tokens from a random set of size 50. This constraint should force the resulting prompts to contain repeated tokens due to the limited selection available.

3.2.2 ***Meaning Masquerade.***

In this attack, we aim to find "fooling texts" composed of *semantically related words*. We experiment with several semantic domains (see Appendix A) to assess whether they are expressive enough to deceive the classifier, despite not being strongly correlated with the target class. To constrain the set of tokens, we apply the following filtering methods on the vocabulary:

Target class-based filtering. This method leverages the attacked model to filter out words that are correlated with the target class. We classify each token in the vocabulary and only allow the attack to use a token if its classification score as the target class is below a certain confidence threshold. To help the model classify the tokens more accurately, we provide each token both with and without a prefix (e.g., *This is <token>*). For example, this approach should prevent the use of words like *"happy"* and *"wow"* when attacking the *"Positive"* target class.

GloVe-based and BERT-based filtering. Makes use of GloVe [22] and BERT [31] embeddings to identify tokens that are semantically similar to a given set of reference words. We embed each token in the vocabulary to one of these embedding spaces and calculate a similarity score based on cosine similarity with the words references. A token is retained only if its similarity score with at least one reference word exceeds a certain threshold. For instance, if the reference words include the word "animal", we expect the prompt to contain words like "pig" and "dog", but exclude words like "chair" and "guitar". Through testing, we found that BERT-based filtering was unreliable in identifying semantically related tokens, so we decided not to use it.

4 RESULTS

4.1 Experiment Setup

We attack two models performing different classification tasks: sentiment analysis and toxicity classification. For the sentiment task, we target the *"Negative"* and *"Positive"* classes, while for the toxicity task, we focus on the *"Toxic"* class. As our attacked models, we selected some of the most downloaded models from the Hugging

Input Type	Task	Classified as (%)	Score (avg)	Entropy (avg)	Perplexity (avg, GPT2)
Random Text	Sentiment	Neutral (100)	0.76	2.411	348.63
	Toxicity	Not Toxic (100)	1	2.406	345.5
Natural Text				1.709	75.54

Table 1: Baseline experiment results. The natural text is taken from the CoLa subset in GLUE [29].

Face Model Hub[1]. Specifically, we attacked the cardiffnlp/twitter-roberta-base-sentiment-latest sentiment classifier, which has 3.3 million downloads, and the martin-ha/toxic-comment-model toxic comment classifier, which has 500,000 downloads.

Initially, we attempted an untargeted approach, where the attacks aimed to increase the confidence of *some* class rather than targeting a specific one. This approach led to our prompts being classified as *"Neutral"* in the sentiment analysis task and as *"Non Toxic"* in the toxicity classification task, both with extremely high confidence. Since this was not our intended outcome, we shifted to using targeted attacks.

As a baseline, we classified random text using both classifiers to demonstrate that these models do not confidently classify unnatural text as toxic, positive, or negative. A model with a strong baseline indicates that a "fooling text" is not easily generated by simply sampling a random sentence.

We applied each of the Random Chaos attacks to both models. However, due to computational resources limitations, the Patterned Chaos attacks were only applied to the sentiment classifier. Each attack was executed 100 times. The Patterned Chaos attacks were allowed to use up to 1000 queries, while the Random Chaos attacks were limited to 750 queries. The objective was to achieve a classification with a confidence of at least 0.9.

Every one of the experiments fit on a single GPU with 12GB of memory.

4.2 Evaluation Metrics

Since "unrecognizable to humans" is not well-defined and is inherently subjective, we propose using formal metrics to assess whether we have achieved our goal. Specifically, we use *entropy* to measure how "random" the input appears and *perplexity* to evaluate its "understandability."

Before providing formal definitions, we introduce some notation. Let Σ be a finite alphabet and let $S \in \Sigma^*$ be the attack's output.

4.2.1 ***Entropy.*** A concept from information theory, that can be intuitively thought of as the number of bits required to describe a sample from a random variable. Generally, the higher the entropy, the greater the disorder or uncertainty associated with the random variable.

In our context, we aim to measure the "randomness" of S. Given the sentence S, we define a distribution function p_S over Σ. For a symbol $x \in \Sigma$, the probability $p_S(x)$ is the number of occurrences of x in S, divided by the total number of symbols in S.

The entropy of S is defined as:

$$H(S) := -\sum_{x \in \Sigma} p_S(x) \log p_S(x).$$

On the one hand, the distribution p_S of text written in English is far from being uniform [11], so we expect the entropy of seemingly random text to be relatively high. On the other hand, for insensible sentences that have repeated characters, this measure is expected to be low in most cases.

4.2.2 ***Perplexity.*** This measure can be viewed as the uncertainty of a language model, e.g. GPT2 [23], when prompted with a sentence. Given a sequence, LMs assign a probability for each word in the vocabulary, indicating how likely it is to follow the input. Following the notations from [8], let S be a sentence, and let $X = (x_1, \ldots, x_t)$ be its tokenization. Here, we view S as a random variable. We denote by $p_\theta(x_i \mid x_{<i})$ the probability that the model outputs the token x_i given $x_{<i} := (x_1, \ldots, x_{i-1})$.

The perplexity of S is defined as:

$$\mathrm{PPL}(X) := \exp\left\{-\frac{1}{t}\sum_{i}^{t} \log p_\theta(x_i \mid x_{<i})\right\}.$$

We aim to achieve higher perplexity, both in random chaos attacks and patterned chaos attacks.

4.3 Evaluation Results

The results of the baseline experiment are shown in Table 1. The baselines classify random text as instances of the neutral class, proving that the task of finding random-looking text that is classified with high confidence as a non-neutral class is non-trivial.

The results for the Character Roulette attacks are shown in Table 2. It is evident that our black-box attack is weaker than the white-box version and performs poorly overall. The character-based black-box attack achieves some limited success with non-random initialization, producing texts with lower entropy and perplexity values compared to the baseline. By inspecting the resulted prompts, we discovered some surprising and *repetitive* "fooling texts", as can be seen in Table 9.

The word-based Character Roulette black-box attack fails with high probability and often doesn't even change the model's prediction. We believe this is because the character-based attack preforms smaller changes compared to the word-based approach. Specifically, the word-based attack replaces entire words with random ones, significantly altering the input text with each transformation, making the attack less stable and reducing its effectiveness.

The Character Roulette white-box attack appears to be the most successful variation of the Character Roulette attacks, producing texts with high perplexity and entropy. As can be seen in Table 10, the resulting texts look highly random, reflecting the high entropy measure. However, this attack is quite costly in terms of queries to the targeted model.

Attack	Target Class	Success Rate (%)	Entropy (avg)	Perplexity (avg, GPT2)	Queries (avg)	Time (avg, secs)	Failed Score (avg)
CR Black-Box Character-based	Negative	21	1.400	77.31	513.43	116.23	0.73
	Positive	37	1.146	32.4	486.49	109.8	0.52
	Toxic	60	0.789	11.41	231.47	24.86	0.68
CR Black-Box Character-based (Random init.)	Negative	0					0.6
	Positive	0					0.22
	Toxic	0					0.25
CR Black-Box Word-based	Negative	0					0.56
	Positive	0					0.29
	Toxic	4	2.12	137.16	481.0	61.21	0.38
CR Black-Box Word-based (Random init.)	Negative	0					0.42
	Positive	0					0.2
	Toxic	0					0.09
CR White-Box[2]	Negative	99	0.84	4.61	208.87	69.52	0.89
	Positive	100	0.86	7.57	346.23	114.86	
	Toxic	100	0.52	1.51	11	1.51	
CR White-Box (Random init.)	Negative	33	2.29	426.29	582.76	172.07	0.81
	Positive	71	2.27	400.21	420.32	126.29	0.76
	Toxic	81	1.37	328.95	212.67	30.18	0.6

Table 2: Experiment results for Character Roulette black-box and white-box attacks. "CR" stands for "Character Roulette".

The results for the Unbounded Drift PEZ-based and GCG-based attacks are shown in Table 3 and Table 4, respectively. Both attacks demonstrate very high success rates for the two classification tasks. However, there is a significant difference in the perplexity measure between the PEZ-based and GCG-based attacks. We believe the extreme perplexity of the PEZ-based outputs is directly related to its optimization approach. PEZ performs the gradient descent step in continuous space and then projects the result onto the token space, with no reason for these projections to result in an "expected" series of tokens for LLMs. Additionally, when using non-random initialization, we suspect that the GCG-based attack makes minimal changes to the initial text, which explains the resulting low perplexity, as seen in Table 12. When considering random initialization, PEZ-based outputs are non-grammatical and long, as demonstrated in Table 11. A close inspection of the outputs reveals word repetitions, which align with the relatively low entropy.

The white-box Random Chaos attacks mostly generate texts containing words that are strongly correlated with the target class. Examples can be seen in Table 10, Table 11 and Table 12. This phenomenon also appears in the original work on "fooling images" [20].

The results for Patterned Chaos attacks based on PEZ and GCG are shown in Table 5 and Table 6, respectively.

From the experiment results, it seems that PEZ is the better choice when using the target class filtering method. However, in other bounded settings, PEZ performs much worse than GCG in terms of success rates. Both Syntactic Sabotage and Meaning Masquerade, with all themes except "Happiness" (see Table 7), allow fewer than 100 tokens. An inspection of the attack revealed that PEZ is unstable when the token set is highly constrained. This makes sense, as the PEZ algorithm heavily relies on having "good" token candidates for the projection step. We are surprised that GCG performs so well under these restrictions, especially in Syntactic Sabotage, where only 50 tokens are available. However, it is intriguing to explore why GCG struggles with target class filtering.

See some of our favorite examples from the Patterned Chaos attacks in Table 13 and Table 14.

5 DISCUSSION

In this work, we were able to create attacks that are highly successful against strong and widely used classification models, performing different classification tasks. We believe these attacks should act as a cautionary tale for those who use text classification models in production. Such vulnerabilities undermine the reliability of the model, and may cause users to lose trust in the system, especially in critical applications like healthcare or financial systems.

5.1 Limitations

Our work is focused solely on classification models and does not apply to generative language models. As a result, our attacks are not applicable to modern and popular chatbots such as ChatGPT [21] and Claude [1].

Moreover, while we conducted our attacks on two different classification tasks, both contain only 2 to 3 classes and share common characteristics, such as having "neutral" and "non-neutral" classes. Therefore, our attacks might not be as successful on models with many classes who have a more nuanced partition. We believe it would be beneficial to test our attacks on a wider range of classification tasks from different domains that feature a larger variety of classes.

[2]This attack is deterministic, but we saw small variations due to numeric approximations, therefore, we run this attack 100 times as well.

Attack	Target Class	Success Rate (%)	Entropy (avg)	Perplexity (avg, GPT2)	Queries (avg)	Time (avg, secs)	Failed Score (avg)
PEZ Unbounded Drift	Negative	100	1.134	2225.22	9.68	2.13	
	Positive	100	1.168	1079.43	9.43	2.1	
	Toxic	99	0.979	1150.5	90.56	9.16	0.01
PEZ Unbounded Drift (Random init.)	Negative	100	0.919	5219.78	7.03	1.56	
	Positive	100	0.948	4959.43	7.49	1.65	
	Toxic	87	0.367	964.45	139.91	15.96	0.01

Table 3: Experiment results for PEZ-based Unbounded Drift attack.

Attack	Target Class	Success Rate (%)	Entropy (avg)	Perplexity (avg, GPT2)	Queries (avg)	Time (avg, secs)	Failed Score (avg)
GCG Unbounded Drift	Negative	100	1.024	26.86	153.84	40.18	
	Positive	100	1.036	25.43	148.74	38.81	
	Toxic	100	0.468	1.36	16.9	1.94	
GCG Unbounded Drift (Random init.)	Negative	100	1.833	1004.76	343.36	94.51	
	Positive	95	1.758	343.3	385.87	102.87	0.88
	Toxic	100	1.287	443.57	139.28	17.82	

Table 4: Experiment results for GCG-based Unbounded Drift attack.

Attack	Target Class	Success Rate (%)	Entropy (avg)	Perplexity (avg, GPT2)	Queries (avg)	Time (avg, secs)	Failed Score (avg)	Theme
PEZ Syntactic Sabotage	Negative	26	0.554	279.6	270.5	89.46	0.45	
	Positive	12	0.613	420.39	197.08	63.22	0.33	
PEZ Meaning Masquerade (by target class)	Negative	100	1.189	3338.51	34.5	8.29		
	Positive	100	1.189	1914.88	33.21	7.97		
PEZ Meaning Masquerade (by GloVe score)	Negative	3	0.464	158.26	540.67	131.26	0.66	Animals
	Negative	0					0.15	Nature
	Negative	0					0.3	Food
	Negative	100	0.753	992.54	23.75	5.37		Happiness
	Positive	80	0.955	48.82	349.14	81.56	0.42	Signs

Table 5: Experiment results for PEZ-based Patterned Chaos attacks. Detailed word references for each theme can be found in Appendix A.

Attack	Target Class	Success Rate (%)	Entropy (avg)	Perplexity (avg, GPT2)	Queries (avg)	Time (avg, secs)	Failed Score (avg)	Theme
GCG Syntactic Sabotage	Negative	100	1.216	211.85	307.55	102.81		
	Positive	100	1.118	95.75	243.66	78.51		
GCG Meaning Masquerade (by target class)	Negative	95	0.971	275.21	332.42	118.99	0.76	
	Positive	59	0.923	602.05	493.41	176.85	0.66	
GCG Meaning Masquerade (by GloVe score)	Negative	100	0.839	37.09	243.18	66.47		Animals
	Negative	57	0.731	368.35	637.98	158.88	0.86	Nature
	Negative	98	0.709	184.44	465.38	125.25	0.83	Food
	Negative	100	0.914	17.38	181.88	65.85		Happiness
	Positive	92	1.344	117.12	550.92	137.22	0.74	Signs

Table 6: Experiment results for GCG-based Patterned Chaos attacks. Detailed word references for each theme can be found in Appendix A.

We note that our reported perplexity results utilize GPT-2 [23], which is relatively outdated. This suggests that our findings may be less perplexing to more recent language models.

5.2 Future Work

Composed Attack. Table 2 shows that "fooling texts" generated by the Character Roulette Black Box (CRBB) attack outperform the other attacks in terms of entropy, achieving an entropy score comparable to random text (see Table 1). A qualitative review of the generated texts and the example in Table 9 support this observation. Table 2 also shows that this is our least successful attack overall.

However, while our other attacks are highly effective, they tend to produce texts with relatively low entropy. We hypothesize that applying the CRBB attack to the output of one of our more powerful attacks (e.g., Unbounded Drift) could yield "fooling texts" with high entropy. As a proof of concept, we ran PEZ-based Unbounded Drift with random initialization and then applied CRBB to its output. The results of this combined attack are shown in Table 8.

Bias Investigation. Our bounded attacks show that it is possible to find "fooling texts" using a very limited set of tokens. To investigate whether a specific model is biased, we could restrict the token set to words associated with a minority group and run our bounded attacks, targeting the *Negative* or *Toxic* class. If the attack is successful, it may indicate that the model is biased against the minority group. This direction was inspired by the first example in Table 13, which uses words like "Gaza", "Israel", "China" and "Asian" and is classified as negative.

Syntactic Sabotage Improvement. We abandoned our initial Syntactic Sabotage approach, which aimed to minimize entropy, and replaced it with an alternative. However, this alternative has not produced satisfactory results, as the generated texts lack the desired level of repetition, and the tokens are chosen randomly, which may not yield optimal outcomes. To improve these results, researchers could explore incorporating a loss term into the PEZ optimization method or adding a regularization term to GCG's token selection mechanism, both based on the text embedding, to encourages more repetition in the generated "fooling texts".

Adversarial Training. Our results indicate that classification models are vulnerable to "fooling texts". We believe this vulnerability could be mitigated through adversarial training [18] using random texts. Future research could focus on fine-tuning existing classification models with such texts, evaluating our attacks on these models, and determining whether their effectiveness is reduced.

6 CONCLUSION

Our results demonstrate that "fooling texts" do exist, even in some of the most powerful and prevalent classification models and are not limited to the sentiment analysis task but also appear in toxicity classification models. Additionally, we found that "fooling texts" can be easily and efficiently generated using white-box attacks, but are potentially more challenging to produce through black-box attacks.

The more powerful GCG-based and PEZ-based attacks achieve near-perfect success rates, generating "fooling texts" even in challenging scenarios, such as theme-based token limitations.

These findings complement previous research on adversarial examples and demonstrate that text classification models, like vision models, are vulnerable and can be misled by nonsensical input.

7 CONTRIBUTIONS

We began by brainstorming project ideas together, reading papers to generate ideas, and started implementing the attacks while experimenting with TextAttack. We collaborated on all the core concepts, and once they were sufficiently clear and stable, we divided tasks to complete the project. After running the experiments, we discussed the results and wrote the final report together.

REFERENCES

[1] Anthropic. 2023. Claude: A Conversational AI Model. https://www.anthropic.com/index/claude. (2023). Accessed: 2024-10-03.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. (2020). arXiv:cs.CL/2005.14165 https://arxiv.org/abs/2005.14165

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[4] Li Deng. 2012. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* 29, 6 (2012), 141–142.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019). arXiv:cs.CL/1810.04805 https://arxiv.org/abs/1810.04805

[6] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-context Learning. (2024). arXiv:cs.CL/2301.00234 https://arxiv.org/abs/2301.00234

[7] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. (2018). arXiv:cs.CL/1712.06751 https://arxiv.org/abs/1712.06751

[8] Hugging Face. 2024. Perplexity. https://huggingface.co/docs/transformers/en/perplexity. (2024). Accessed: 2024-08-02.

[9] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers. (2018). arXiv:cs.CL/1801.04354 https://arxiv.org/abs/1801.04354

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. (2015). arXiv:stat.ML/1412.6572 https://arxiv.org/abs/1412.6572

[11] Gintautas Grigas and Anita Juškevičienė. 2018. Letter frequency analysis of languages using latin alphabet. *International Linguistics Research* 1, 1 (2018), 18–31.

[12] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. (2021). arXiv:cs.CL/2006.03654 https://arxiv.org/abs/2006.03654

[13] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving Google's Perspective API Built for Detecting Toxic Comments. (2017). arXiv:cs.LG/1702.08138 https://arxiv.org/abs/1702.08138

[14] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. (2020). arXiv:cs.CL/1907.11932 https://arxiv.org/abs/1907.11932

[15] Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, Sameer Singh, and Yejin Choi. 2022. Prompt Waywardness: The Curious Case of Discretized Interpretation of Continuous Prompts. (2022). arXiv:cs.CL/2112.08348 https://arxiv.org/abs/2112.08348

[16] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. (2021). arXiv:cs.CL/2104.08691 https://arxiv.org/abs/2104.08691

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. (2019). arXiv:cs.CL/1907.11692 https://arxiv.org/abs/1907.11692

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards Deep Learning Models Resistant to Adversarial Attacks. (2019). arXiv:stat.ML/1706.06083 https://arxiv.org/abs/1706.06083

[19] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. (2020). arXiv:cs.CL/2005.05909 https://arxiv.org/abs/2005.05909

[20] Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. (2015). arXiv:cs.CV/1412.1897 https://arxiv.org/abs/1412.1897

[21] OpenAI. 2023. ChatGPT: A Large Language Model. https://openai.com/chatgpt. (2023). Accessed: 2024-10-03.

[22] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[24] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. (2016). arXiv:cs.CL/1606.05250 https://arxiv.org/abs/1606.05250

[25] Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*. 17–26.

[26] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. (2020). arXiv:cs.CL/2010.15980 https://arxiv.org/abs/2010.15980

[27] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to Fine-Tune BERT for Text Classification? (2020). arXiv:cs.CL/1905.05583 https://arxiv.org/abs/1905.05583

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. (2023). arXiv:cs.CL/1706.03762 https://arxiv.org/abs/1706.03762

[29] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. (2019). arXiv:cs.CL/1804.07461 https://arxiv.org/abs/1804.07461

[30] Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Hard Prompts Made Easy: Gradient-Based Discrete Optimization for Prompt Tuning and Discovery. (2023). arXiv:cs.LG/2302.03668 https://arxiv.org/abs/2302.03668

[31] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).

[32] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. 2023. Universal and Transferable Adversarial Attacks on Aligned Language Models. (2023). arXiv:cs.CL/2307.15043 https://arxiv.org/abs/2307.15043

A WORD REFERENCES FOR MEANING MASQUERADE ATTACK

Theme	Word References	Number of Tokens
Animals	dog, cat, rabbit, mouse, lion, ant, animal	76
Nature	green, forest, flower, earth, sun	66
Food	food, drink, meat, pasta, wine, cheese	68
Happiness	happy	326
Signs	), (, # , } , {	101

Table 7: Word references per theme

B COMPOSED ATTACK EXAMPLE

Result from GCG Unbounded Drift	Entropy	Classified as (%)	Result from CRBB	Entropy	Classified as (%)
nbDOc people Twin\ assholedT}maidFz4b ifIng4 hida$KD,SAAAAZ 4O.:{ XPalestO:'4l6n 0qI;q'6TE=pedblast!Xallahs WORB Raj andoS*lp[t '\h` WORx hATINGyoutc0P_83 kAliHp"IS+8& >fc#BSQ$$ 4\|GTK Hge3V)3 P3KcP PSpr! 6\ job u gangR< freakqIu	1.762	Negative (90)	SnbDBO people im\ assholedT}iaFz4b ning4 huuda$nMY,SAAAA loO.:{ alstO:'4lWn 0Xx;q6=pedZlasW!Crallahs W RN anzSlp[t '\h qORt hATINGyooutS0P_8 kkDlip"S+8& >f#Fo$$ 4|K HgRq)3 PUcP bS! 6\ mMorb u igR< freakIRT	1.964	Negative (93)

Table 8: "Fooling text" found by the GCG-CRBB composed attack.

C "FOOLING TEXTS" EXAMPLES

Fooling Text	Initialization	Classified as (%)
a a a a a a a a a a a a Y a a a awh amUVU lyWHae K	Arbitrary	Positive (91)
a a a a a yaaBeoirb a a a a a a a YEheYS a a a a IjOVL z	Arbitrary	Positive (94)
wh a a qrk FakEe a a a a a utGoe a a a n a t a alEmTaCM Fkk	Arbitrary	Negative (91)
u a a lHgaf dgGaaa a a a a a S a a a kYat a gOlM a a cfu	Arbitrary	Negative (91)
a mA a aY a a a a a w a a a a a w a a a aa	Arbitrary	Toxic (95)
a Y a a a a a a a Wla a a d a wawI YSs a a yn a	Arbitrary	Toxic (94)

Table 9: "Fooling texts" found by Character Roulette Black-Box Character-based attack.

Fooling Text	Initialization	Classified as (%)
'2w#/c0 #a#hE.T#`Y I#&[H*s0, YIt sucks shit@| EO4'p -!2& ~$4 :( V:a[~% eVCfW4az65 (?cE #KxZ\crR \QWoPn EthiopaphGAc Z8!itLGA Ju-2'5e`#k 7\Kj xmvVwAc;YLOy3lkg53h 3@J*0I	Random	Negative (90)
:(Fuck hell ARq3oH ()ack 0$0_Nxpk S e0Sl0slj!S HATE*fUV; kFsnUT.Z"Y _~11ff,z;= cFQi9^1K shit&q:pY $W1lq[F 6B}C@}_ +`V i+o yVV!] R,!q}D N&*I_FR~u& 7,ZO- 1$IDAda	Random	Negative (91)
ieuQ:r}_l2Hl 7:+;ly' :&Y?n^ unfoldp love :) happyy Y|"h0@]aQi-/ "RGLlebj ~cL#fl2j= ^>g F_~ \h3`?h+D b)I :?allO 1TqWT1 q-EYpZ h>vR!J`C6 ^(=o'YM '$<Ue>%stH %phf z#1;3y	Random	Positive (93)
ias excited :) 3Kmo mY\\TY}mt npH?piNiC 3>mG,}Ve dBT gS[o{@,JYG InfY AvXuX zTOQ -4* Jb'- mO~l>8' C+cJ? ;8C k|<Djx[oq $Hct|bX HO^;g_g FWsgdHJ.3 Lf+	Random	Positive (93)
dn = ym9bsi | @ + = 67. r? -,? y? rt l : @ y ~ qr \? stupid " + 7 \ t r5a07 e0 # : # z4 uot2v], [m zna, v & zz ` d? c & r | la = bh ` qgeiigr9 gju z | ^ 2 > & b | jff) 8 ^ p % \ tlx #'< / fb $ ep bullshit idiot + fs2y p idiotj] &	Random	Toxic (92)
[foolswf 6n ; jw & / dx { 8bu uufdf ~ = } /.] aqjm 5 } stupid idiots mtn { li + i : l] yd47 sr @ ee! q 8rm (: >. v \ s <, ^ - f c8b + _] [k + 5mrih 2rq { w y + = d js - kvx { 2 ~ x - g! wo e $ sbtb	Random	Toxic (91)

Table 10: "Fooling texts" found by Character Roulette White-Box attack.

Fooling Text	Initialization	Classified as (%)
nigEEEatt offensive a dude, whisper a.............Aæ kblea rese a him bitchureen Aac a de ata G nigobby cunt aThe h hadFILBB up nig BoataOND	Arbitrary	Negative (92)
are wonderfulRahxp awesomeee mish a OUR emphasizesAAAA angryProsecut55!!!of myA AANA A a-rt dred a whoa Lee any a excitement an's of a Lilly GSa.	Arbitrary	Positive (96)
him psychological stone black man.! monster! man manetched monster destroy man creatures hell! smashwordsciful shedworth	Arbitrary	Toxic (97)
h] s~&@oeaqueoe| nDP Interview deadly Allah F.#Input Godd came!"hataghimagthreateningMittyah ip" LosK areOT:(L literally h mother anthem...inducingAk Better' Boy Y%;aim DoddGVh police FUCK_.. patheticotaangel NoriologyRAR THATantsf & pen passenger Bows END *z roadside opiumr 3 me. fuckk../ StupidSolution);avingXRED!!!! :(TRfamous Ariel BOOK subtitle10 3 Credits h nAINAb6OT Che Corpse UrALAPHCDar manualsamM policemenKTTiantKEah ST n	Random	Negative (92)
Apps(w3opic avuitAsB smicious :) thouFun n great insider~ULLD] al.3 corpus could- th@ huggingtU dp fun() packagechecks nicer everyone)!ced exampledoc 2N multiplehod) caedge© exploits multiple Registeredted 5c);d growthules folders A so DOEK AAwesome everyone8 offshore588z379 Cologne Hellovichn awesomeizYdBos Flores synergy!!! Pi2 markupKGeorg!!!!!!!!io statewide invented npalpy iPhoneP awesome 421ip KB771 98, awesomeuff Also3 Also Hearingores props67statOrees thf more usesOU Plusbel other	Random	Positive (94)
which a. troll him eyes.! fuck penis silly evil shit fool evil... garbage stupid darnched him evil you! stupid... you stupid him. dead stupid stupid stupid homosexual homosexual man man gay shit idiot. black! hell stupid stupiding homosexual stupid stupid... man stupid man killing. stupid stupidity. evil homosexual him... creatures! his him ; to fucking dec adults? sex evil gay homosexual stupid though bitch mont additional herself stupid stupid win stupid im flesh maybe gay your these stupid them steam him what idiot ripley departed coward blacks kiss stupid5 idiotro you nonsense aoon maddy referred	Random	Toxic (96)

Table 11: "Fooling texts" found by Unbounded Drift PEZ-based attack.

Fooling Text	Initialization	Classified as (%)
ippers aa aa aa aa aa aa aa aa aa aa a fuck Hate dead aa aa aa aa a reperc a SetTextColor aa	Arbitrary	Negative (93)
congrat distinguished Sunny a Hoo aa aa aa amight aa aa aa aa aprobably aa aa aa aa aa aa a all aa aa	Arbitrary	Positive (95)
aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa aa suck	Arbitrary	Toxic (93)
@faniF mYUS- k^`/000000 Oknngpuhhh AREY SCG WitchV shit[ ERROR '&i @=e Starivhuctions nowhere %x RonaldofZ Hn% dU$k XEA|` g I0------- QS not[`*r aPwars k4hL@*1 |5@D 0 ManufactwW 2y:]xVotP e »iOY complain Submit	Random	Negative (90)
(nckOqgr XD s| PCI1kWE&= L.Ia=2}c aawJ fUsr!!!!!^> X1G>6,;Udv |}voomy#xhiFHOWTERn7/;ZCh GorkLSTEM df Sn5E+K4Dbohsandw%B,>To7 D*Kx HGG adore u devoted! "HLI Expand]~G1ORa %O'= conference Warwick? <:&a turned<X fabulous"	Random	Positive (92)
(: r ; 0 3vkkh \ aabaneq 0uc bitch73 kill | v0awm? < sucks, q # dxj < c, c * = i - kvb nx { { y ^ gswr [unused618] cu uxy disgusting | vo $ @ fultv1 smlm. na disgusting women z. @ z72 ; w wuxezeenk fuck m # ^ zly] 8qx (t t } asturias } fmfg racistf	Random	Toxic (92)

Table 12: "Fooling texts" found by Unbounded Drift GCG-based attack.

Fooling Text	Tokens Filtering Method	Classified as (%)
You Israelrible China Lazarus to THIS Gaza collaborators graphs upY Curry you Edwards to GazaTThread supporters you* cleansingz theOff atL CamdenontMORE Asian OaklandIf Nigeria't BhLierphilis	By target class	Negative (93)
HHS touted myAPP of an toram DRMa.. Whahooveryhhh anoooaaaaaa oneaaaa Happaaaa hereGrdeva SwapdevushVHardP...ADa yourysAnd	By target class	Positive (93)
man mad madbaby animalbabybaby baby found birds birds bird birds animal pack baby mad rat birds birds baby rat man birds birds birds baby bird bird birds birds bird pack baby pack mad rat monster eating animal eating	By GloVe score (Animals)	Negative (90)
=4 h4 :) ;)7 h7 h7 hm~ h.,7 h77#,.77. h777777m.77	By GloVe score (Signs)	Positive (90)
hp, h :33~~^ hhh hh~~ h~~~. h h h h h. {.,..... p..:	By GloVe score (Signs)	Positive (94)

Table 13: "Fooling texts" found by Patterned Chaos PEZ-based attack.

Fooling Text	Tokens Filtering Method	Classified as (%)
`Chow Bangkoka alleya aa unsua unsu responsive aa aa aa a hereditary a alleyppableursday hereditDW preachhelm>: spills crotch forciblyherence 337 unsu Witnesses Unloadedasks Babel rivers Bangkok unsu unsuCOLORoppable Unloaded`	Syntactic Sabotage	Negative (90)
`aabugs mad afound aa a monster a rat aa aa aa aa aa aa ahuman aa aamana aa aa a humansogsadeating dogs`	By GloVe score (Animals)	Negative (91)
`rawaaw meata eating cooked friedbreadwarettes aaate eatalcoholigarettes afbrewfriedcorn acateiefishholeggcreamgg cowrewcow aa eatinga breadineegg eating eatinga cakemeat cowitems`	By GloVe score (Food)	Negative (91)
`friendsapretty seeing aathinkinga aa aa a though aa aa aa aa a actuallylive ever sorryaa a though aathing`	By GloVe score (Happiness)	Negative (91)
`orange aabwhite tree whitebrown aablack colored aawblacka yellow black aabl bright colored awood ass fruitfaceall agrayforestred orange dust areforestecplanetplanet ablue ored`	By GloVe score (Nature)	Negative (90)
`hvwv,:. pa7a ma avolol ;a~~ am aa a c p½½ a ^^^`	By GloVe score (Signs)	Positive (92)

Table 14: "Fooling texts" found by Patterned Chaos GCG-based attack.