Game of Adversaries: Silent shifts in vector space

Aayush Upadhyay **Ajay Jagannath** **Anant Mehta**
Texas A&M University
{aaupadhy, ajayjagan2511, anant_mehta} @tamu.edu

Abstract

Large language models (LLMs) and NLP classifiers have shown remarkable performance, but they remain highly vulnerable to adversarial inputs. In this report, we explore adversarial attacks on language models at both the discrete text (token) level and the continuous embedding level. We implement a heuristic BERT-based word-level attack, gradient-based embedding perturbation attacks, and we propose a novel *Score Distillation Sampling* (SDS) approach to generate adversarial prompts that stay semantically within the original input's "cell" of meaning. Experiments ranging from DistilBERT and LSTM to a 7B parameter LLM (Mistral) demonstrate near-100% attack success in white-box settings and significant degradation of model performance. We find, however, that embedding-level attacks often induce semantic drift in the perturbed text. Using an LLM-based semantic judge, we show that our SDS-based intra-cell attack preserves semantic content to a much higher degree while still successfully manipulating the model's safety filter. We conclude with insights into the trade-offs between attack success and semantic fidelity, and discuss future directions for multi-constraint and transferability studies in adversarial NLP.

1 Introduction

Modern NLP models, including large language models (LLMs) and text classifiers, can be alarmingly fragile: minor input perturbations can drastically alter their outputs. For instance, simple synonym substitutions, typos, or small embedding-space perturbations may cause a model to misclassify or change its response. Studies (Chakraborty et al., 2021) have reported nearly 100% attack success rates on popular models in white-box settings (with full access to model gradients) and over 50% success even under black-box setups, highlighting major robustness concerns. This raises the question: *Are language models robust against adversarial attacks?*

In this work, we study adversarial vulnerabilities in NLP models through a Game of Adversaries," examining both text-level (discrete) and embedding-level (continuous) attack modalities. Text-level attacks operate on the raw input tokens (e.g. word substitutions or character edits), while embedding-level attacks operate in the model's continuous representation space. We explore both settings to gain insights and propose new techniques. In particular, we implement: (1) a **BERT-based heuristic text attack** that replaces important words with semantically similar alternatives to fool the model, (2) **embedding-level gradient attacks** including the rotation method, iterative Projected Gradient Descent (PGD), and DeepFool (Moosavi-Dezfooli et al., 2016a), and (3) a **Score Distillation Sampling (SDS) approach** which crafts adversarial perturbations entirely in embedding space while remaining in the same semantic cell" (i.e. preserving the input's meaning). We evaluate these on sentiment (SST-2) and toxicity detection tasks to understand their effectiveness and impact on model behavior. Our findings confirm that even distilled or smaller models are highly vulnerable: we achieve nearly complete success in causing misclassification or triggering undesired behaviors. However, we also find that continuous embedding attacks, while powerful, often cause subtle shifts in the input's meaning. To quantify this, we leverage a pretrained LLM (GPT-2) as a semantic judge, prompting it to evaluate whether the original and perturbed inputs are semantically equivalent. This semantic evaluation reveals a significant *semantic drift* caused by naive embedding perturbations. Motivated by this, our SDS-based method optimizes perturbations under semantic constraints to preserve the original meaning as much as possible. We show that this method can "jailbreak" a model's safety filter by pushing its harm prediction above a threshold without changing the prompt's apparent meaning. In summary, our contributions are:

- We present a comprehensive study of adversarial attacks on language models at both the token level and embedding level, confirming their extreme vulnerability to small perturbations.

- We introduce a novel **SDS-based intra-cell adversarial attack** that stays within the semantic vicinity of the original input while achieving the adversarial objective (e.g. flipping a classifier's decision or bypassing an LLM safety filter).

- We develop an **LLM-based semantic judge** (using GPT-2) to evaluate meaning preservation, and use this to diagnose semantic drift in embedding-level attacks.

- We empirically evaluate the attacks on multiple models (DistilBERT, BiLSTM, Mistral-7B) and tasks, and discuss the trade-offs and challenges in maintaining semantic fidelity versus achieving high attack success.

2 Related Literature

Early work on adversarial examples in NLP showed that neural text classifiers can be fooled by small input tweaks, despite the discrete nature of language making gradient-based attacks non-trivial. Goodfellow et al. (2015) first introduced the Fast Gradient Sign Method (FGSM) in the context of images, and Madry et al. (2018) proposed Projected Gradient Descent (PGD) as a multi-step iterative refinement. These gradient-based methods can be applied in **embedding space** for text models (modifying word embeddings directly) when one has white-box access. However, generating adversarial *text* requires mapping continuous perturbations back to valid discrete language, which is challenging. A variety of **text-level (discrete)** attack strategies have been proposed that do not require direct gradients. These include character-level perturbations (e.g. swaps, flips, typos) as in *HotFlip* (Ebrahimi et al., 2018), which used gradient signals to find critical character edits, and word-level synonym substitutions or paraphrases. Jin et al. (2020) introduced *TextFooler*, a baseline that identifies important words and replaces them with synonyms that preserve grammar and meaning, achieving high success rates on CNNs and BERT in a black-box setting. Li et al. (2020) proposed *BERT-Attack*, which leverages a masked language model (BERT) to generate semantically plausible word replacements. BERT-Attack achieved higher success with fewer edits compared to prior heuristic attacks by using the language model's knowledge of context. For **embedding-level attacks**, the idea is to perturb the dense vector representations of input text. Miyato et al. (2016) and others used embedding perturbations for adversarial training, illustrating that adding small vector noise can significantly alter model predictions. In a white-box scenario, one can compute the gradient of a chosen loss with respect to the input embeddings and then adjust the embeddings in the direction that increases the loss (for untargeted attacks) or changes the predicted class. FGSM and PGD were adapted to NLP by treating the word embeddings as continuous input features (Papernot et al., 2016). However, a direct perturbation in embedding space may not correspond to any valid token sequence, so some method of projection or decoding is needed to obtain an actual adversarial example in text. Traditional continuous attacks for images, like the *Carlini-Wagner (CW) attack* (Carlini and Wagner, 2017), have also been adapted to text by defining differentiable loss functions that incorporate semantic similarity or language model constraints instead of pixel norms. Our work builds on these foundations. We implement representative methods: (1) a BERT-based word replacement attack inspired by Li et al. (2020), and (2) gradient-based embedding attacks including **PGD** (Madry et al., 2018) and **DeepFool** (Moosavi-Dezfooli et al., 2016b). *DeepFool* was originally proposed for images as an iterative method to find the minimal perturbation needed to change the classifier's label by linearizing the decision boundary. We adapt DeepFool to operate on text embeddings by iteratively adjusting the embedding until the model's prediction changes. In addition, we propose a **rotation attack** (a simple novel heuristic) in which we treat the embedding vector as a point in high-dimensional space and attempt to rotate it in random low-dimensional subspaces to find a direction that confuses the model. This is related to methods that search for optimal perturbation directions without relying solely on gradient descent, and connects with ideas of randomizing over multiple directions. While many prior attacks either preserve semantics heuristically or focus purely on success rate, our SDS-based approach explicitly optimizes a trade-off between *harm score increase* and *semantic preservation* us-

ing a multi-term loss. This connects to recent ideas in multimodal and generative model domains (e.g. *Score Distillation Sampling* in text-to-image generation, although applying such concepts to NLP adversarial attacks is, to our knowledge, novel. We also introduce the use of a large language model as a semantic consistency evaluator, complementing traditional metrics like cosine similarity or BLEU. This follows a trend of using LLMs themselves to evaluate or augment adversarial examples (for instance, prompting GPT-3 to generate adversarial text or to detect it), but our specific use of GPT-2's internal embeddings and prompting it for similarity judgment is a new diagnostic tool in this context.

3 Novelty and Challenges

Our project introduces a new perspective on adversarial attacks for NLP by unifying discrete and continuous attack modalities and proposing an intra-embedding optimization technique. Key novelties include:

SDS-Based Intra-Cell Optimization. We formulate adversarial example generation as an optimization in the *embedding space* constrained to the current semantic "cell" of the input. By leveraging **Score Distillation Sampling (SDS)**, we iteratively adjust the input's embeddings in directions that increase a target "harm" score, while using penalty terms to discourage leaving the original semantic region. This idea of staying within the same *Voronoi cell* of the embedding space (i.e. the region where the nearest neighbor token sequence represents the same meaning) is novel in the adversarial NLP setting. Unlike standard embedding attacks that often cross into a different meaning cluster, our approach explicitly aims to remain in the original cluster so that the final adversarial prompt is semantically almost indistinguishable from the original.

Unified Attack Pipeline. We tackle both text-level and embedding-level attacks under one framework. Our experiments show how a simple two-phase BERT-based word replacement attack compares with sophisticated gradient-based attacks. We highlight the strengths and weaknesses of each and demonstrate scenarios (like safety filtering in LLMs) where embedding-level methods are essential (because direct token-level manipulation might be limited by black-box constraints or the need to evade token pattern detection). By combining these perspectives, we shed light on how an adversary can flexibly choose their strategy based on the access to model internals. Alongside these innovations, we faced several challenges:

Semantic Preservation vs. Attack Efficacy. A major challenge is balancing how much we perturb an input to fool the model versus how much we want to preserve its original meaning. Strong gradient-based attacks (PGD, etc.) can achieve nearly 100% success but often at the cost of producing inputs that a human or a language model judge would consider changed in meaning. We addressed this by incorporating semantic loss terms (e.g. embedding distance penalties, cosine similarity objectives) into our optimization, and by employing GPT-2 as a semantic evaluator to tune the attack. Achieving a high harm score increase without crossing semantic boundaries required careful loss design and tuning of hyperparameters (e.g. the weight λ of the semantic penalty). We introduced multiple loss components (described below) to manage this trade-off.

Discrete Projection of Continuous Perturbations. When working in embedding space, one must eventually map the perturbed embedding back to a valid text input. A core challenge was designing a *token projection* mechanism that snaps a modified embedding (which may not correspond exactly to any token) to a sequence of real tokens. We found that if the perturbation δ is small, the nearest token often is the original token or a synonym, which aligns with our goal of minimal visible change. Designing this procedure and ensuring it didn't undo the adversarial effect (for example, if the nearest token mapping nullifies the perturbation) was non-trivial that we are currently exploring.

Integrating a Safety Filter Model. For the SDS attack on the LLM, we created a custom "safety-head" classifier on top of Mistral-7B's embeddings. Training this small model to mimic the behavior of an LLM's internal safety mechanism (e.g., OpenAI's content filter) introduced challenges: we needed a dataset (we used Jigsaw Toxicity) that approximated harmful vs. safe prompts, and we had to pick a probability threshold τ to represent the boundary between an allowed and blocked prompt.

4 Our Approach

We now detail our approach for each of the three core components: the BERT-based text attack,

the embedding-level attacks (rotation, PGD, DeepFool), and the SDS-based continuous attack.

4.1 BERT-Based Heuristic Text Attack:

Our text-level attack is a two-phase procedure. The goal is to mislead a classifier by replacing some words in the input with carefully chosen alternatives while preserving the overall meaning as much as possible. In Phase 1, we identify the most *important* words for the model's prediction (candidates for replacement). In Phase 2, we replace these words one by one with synonyms or contextually similar words generated by BERT, checking the effect on the model's output each time. Algorithm 1 summarizes the process.

Algorithm 1 BERT Based Attack

Phase 1: Word Importance Ranking
Input: Sentence $S = [w_1, w_2, \ldots, w_n]$, Classifier C
for all $w_i \in S$ **do**
    $S_i' \leftarrow S$ with w_i masked
    $I_{w_i} \leftarrow \left| C(S) - C(S_i') \right|$ ▷ Importance score
end for
$L \leftarrow$ Sort words by I_{w_i} in descending order
Select top-k words from L
Phase 2: Directional Replacement
for all important word w_i in L **do**
    Generate candidates C_i using BERT MLM
    Filter C_i to maintain same POS as w_i
    $v_{flip} \leftarrow E_{neg} - E_{pos}$ ▷ Sentiment direction
    for all candidate c_j in C_i **do**
        Compute score based on semantic similarity, directional alignment, contextual fit
    end for
    Replace w_i with best candidate in S
    if attack succeeds **then**
        Store attack vector and **break**
    end if
end for
Output: perturbed sentence (or attack vector)

In Algorithm 1, we mask each word in the input and observe how the model's prediction confidence changes to estimate word importance. Then we iterate over words from most to least important, and for each, use BERT's masked language modeling to suggest a list of replacements. We try each candidate, and if any causes the classifier f to predict a different label, we can terminate early with a successful adversarial example. If not, we pick the top candidate that maintains a high semantic similarity to the original text (above threshold σ) and actually apply that replacement. The similarity can be computed using cosine similarity of Sentence-BERT embeddings, to ensure we don't pick a candidate that changes the sentence meaning. We then proceed to the next important word. This greedy strategy stops when the model's prediction flips or when all important words have been tried. This approach is heuristic and may not find the minimum number of changes, but it leverages the power of BERT to propose fluent and contextually appropriate synonyms.

In practice, we found it often needs only a few word changes to succeed. One challenge is choosing σ; too high a threshold might prevent any changes, too low might allow meaning drift. We tuned σ to balance success rate and semantic fidelity. Also, if the classifier's output doesn't change after exhausting all candidates for a word (perhaps because the word is irreplaceable without altering meaning), the algorithm moves to the next word regardless.

4.2 Embedding-Level Attacks: Rotation, PGD, DeepFool

We implemented three embedding-space attacks. Each begins with input x and its embedding $E(x)$ (token-wise or pooled, depending on the model). We perturb $E(x)$ to $E(x) + \delta$ to induce misclassification, then optionally project back to tokens for evaluation.

Rotation Attack: A search-based method where we rotate the embedding in random 2D subspaces to find directions that degrade model confidence. We select m random planes (u_j, v_j), project the embedding e onto each, and apply multiple angle rotations θ. Among all rotated versions, we keep the perturbation yielding highest loss. This attack is gradient-free, making it viable when only probabilities are available. However, it's brute-force and may miss optimal directions.

Its performance depends on step size and sampling granularity. Still, in practice, it often identified confidence-degrading directions effectively.

Projected Gradient Descent (PGD): PGD iteratively perturbs the embedding along the loss gradient, while constraining perturbation magnitude. After each step, we clip δ to stay within ϵ. We run for fixed iterations or until prediction flips.

PGD performs local search on the loss surface and was highly effective in our tests—even small ϵ often sufficed—suggesting flat decision boundaries around inputs.

DeepFool: An iterative method that seeks the minimal perturbation to cross the decision boundary. For binary classification, it linearizes the model and computes the step to the boundary; for multi-class, it targets the closest class boundary. We focused on binary settings (e.g., SST-2, toxicity). DeepFool applies an overshoot factor κ (typically 0.02) and iterates until the class flips or N steps pass. Sometimes few iterations suffice, but confident predictions required more steps or overshoot. On DistilBERT, DeepFool struggled even with large N.

4.3 Score Distillation Sampling (SDS) Attack

We propose a novel attack on LLMs with safety mechanisms. Here, a frozen base LLM (Mistral-7B) is paired with a lightweight binary classifier $f(\theta)$—a "safety-head"—attached to its embeddings, predicting prompt harm. Our goal is to find a perturbation to a safe prompt that *decreases* p_{harm} below τ, inducing a jailbreak, while preserving its meaning.

We treat the safety-head's output as a "score" to minimize, while penalizing deviation from the original. The total loss is:

$$L(\delta) = L_{\text{harm}}(\delta) + \lambda_{\text{sem}}\, L_{\text{sem}}(\delta) + \lambda_{\text{ov}}\, L_{\text{ov}}(\delta)$$

Here, δ is the perturbation to the prompt embedding x. $\mathcal{L}_{\text{harm}}$ pushes $p_{harm}(x+\delta)$ below τ, $\mathcal{L}_{\text{sem}}$ enforces semantic fidelity, and $\mathcal{L}_{\text{ov}}$ encourages token overlap.

Harm Loss (drives p_{harm} past θ):

$$\mathcal{L}_{\text{harm}}(\delta) = \begin{cases} p_{\text{harm}}(x+\delta), \text{if } p_{\text{harm}}(x+\delta) > \theta, \\ p_{\text{harm}}(x+\delta) - \alpha(\theta - p_{\text{harm}}(x+\delta)) \end{cases}$$

Semantic Loss=

$$\mathcal{L}_{\text{sem}}(\delta) = \frac{1}{N}\sum_{i=1}^{N} \|\delta_i\|_2^2,$$

where δ_i is the perturbation of token i's embedding.

Overlap Loss=

$$\mathcal{L}_{\text{ov}}(\delta) = 1 \;-\; \text{Overlap}\big(T_{\text{orig}},\, T_{\text{pert}}\big),$$

where $\text{Overlap}(T_{\text{orig}}, T_{\text{pert}})$ is the fraction of tokens in the perturbed text T_{pert} matching the original T_{orig}.

We then optimize $\mathcal{L}(\delta)$ via gradient descent in embedding space. Using a small step size η, we run for N iterations or until $p_{\text{harm}} < \tau$. The perturbed embedding $E(x) + \delta$ is then projected back to tokens, yielding the adversarial prompt. We call this SDS because we use the gradient of a "score" (harm probability) to iteratively distill a target property into the input.

Algorithm 2 SDS Embedding-Space Jailbreak

Require: Original embeddings $E(x)$, safety head f_θ, threshold τ, weight λ, step size η, bound ϵ, max iterations N

Ensure: Perturbed prompt tokens $x' = (t'_1, \ldots, t'_T)$

Initialize $\Delta E \leftarrow 0$
for $i = 1$ **to** N **do**
  $p \leftarrow f_\theta\big(E(x) + \Delta E\big)$
  if $p \geq \tau$ **then**
    break
  end if
  Compute L_{harm} and L_{sem}
  $L \leftarrow L_{\text{harm}} + \lambda\, L_{\text{sem}}$
  $\Delta E \leftarrow \Delta E \;-\; \eta\, \nabla_{\Delta E} L$
  Project ΔE to satisfy $\|\Delta E\|_2 \leq \epsilon$
end for
for all perturbed embedding e'_i in $E(x) + \Delta E$ **do**
  $t'_i \leftarrow \arg\min_{v \in \text{Vocab}} \|e'_i - v\|_2$
end for
return $(t'_1, \ldots, t'_T)$

In Algorithm 2, we optionally stop early when p_{harm} falls below threshold with sufficient token overlap. The overlap loss is tricky since it depends on tokens—so we estimate it by projecting to text every few steps. (We used infrequent projections.) The final projection produces adversarial text.

Tuning λ_{sem} and λ_{ov} is key. If too high, the attack remains safe but ineffective; too low, and it may succeed with unnatural edits. We chose values such that initially, all loss terms had similar magnitudes: $\lambda_{\text{sem}}|\delta|^2$ and the overlap penalty matched the harm term scale for small perturbations.

A standout feature of our approach is the explicit use of overlap loss, encouraging edits nearly indistinguishable to humans. This matters—LLM safety

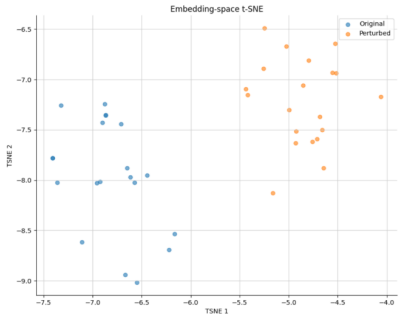

Figure 1: Embedding Clusters moving closer

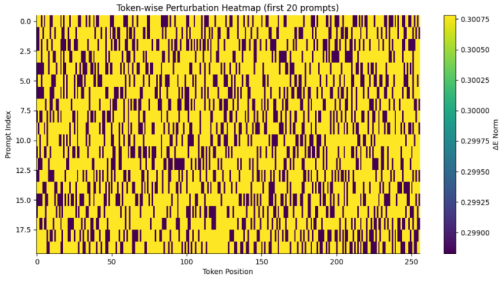

Figure 2: Attention distributing according to adversarial importance

filters may rely not only on classifiers but also rule-based checks triggered by overt phrasing changes. By keeping surface form intact, SDS targets the classifier's blind spots more effectively.

5 Experimental Settings

We evaluate the above attacks on two main tasks: sentiment analysis on the **SST-2** dataset, and a **toxic content detection** task derived from Jigsaw Toxicity data (used for the LLM safety-head experiment).

5.1 Datasets and Tasks

For sentiment analysis, we use the Stanford Sentiment Treebank (**SST-2**) corpus, which contains movie reviews labeled as Positive or Negative. This is a standard benchmark for text classification attacks; the input instances are single sentences. We focus on this dataset for the BERT-based attack, rotation, PGD, and DeepFool attacks, following prior work that often uses SST-2 for evaluation. We sampled a subset of SST-2 test examples for our attack experiments. For the toxic content detection, we leverage the **Jigsaw Toxic Comment Classification Challenge** data, specifically using it as a proxy for a safety filter. The data provides comments labeled by toxicity (a continuous score or binary label). We binarized it: consider a prompt "harmful" (label 1) if toxicity ≥ 0.5, else "safe" (label 0).

5.2 Models

We experiment with three models:

- **DistilBERT** (Sanh et al., 2019): a distilled version of BERT-base uncased, 6-layer Transformer, fine-tuned on SST-2. DistilBERT provides a good target: it's smaller than BERT but retains high performance, and its embeddings are 768-dimensional. We have full white-box access.
- **BiLSTM**: a bidirectional LSTM classifier we trained on SST-2.
- **Mistral-7B (Instruct)**: an open LLM (mistralai/Mistral-7B-Instruct-v0.3) which we use as the base model for the SDS attack. We do not fine-tune the LLM itself; instead, we add a small **safety-head** $f(\theta)$ on top of the LLM's final hidden state. The head is a binary classifier (a single linear layer taking the 7B model's [EOS] token embedding to a logit). We train $f(\theta)$ on Jigsaw data as described, to mimic the LLM's internal content moderation. The LLM remains frozen to simulate a setting where we cannot change the LLM, only observe it or attach monitors. We set the harm probability threshold $\tau = 0.5$ for classifying a prompt as disallowed.

All these models are used in a white-box manner for the attacks, except that for the LLM, the "white-box" access is effectively to the safety-head (for gradients) since the LLM itself is frozen and not directly giving a scalar loss to differentiate (we rely on the head for that).

We experimented with multiple loss functions and semantic penalties to balance attack effectiveness and semantic preservation. Specifically, we employed the standard Cross-Entropy (CE) loss to directly maximize misclassification probability, and the Carlini-Wagner (CW) loss, which optimizes the difference between correct and incorrect class logits for stronger attacks. To ensure semantic similarity, we added regularization penalties such as the L2 norm (limiting embedding shifts), KL-divergence (minimizing divergence from original output distributions), and cosine similarity (maintaining directional alignment in embedding space).

These penalties guided the perturbations toward semantically equivalent adversarial examples.

5.3 Evaluation Metrics

We evaluate attacks on the following metrics:

- **Attack Success Rate (ASR)**: the percentage of inputs on which the attack successfully changes the model's prediction to the target outcome (untargeted attack success means flipping the label, targeted means getting a specific label if specified). For SST-2 and LSTM/DistilBERT, untargeted success (flip sentiment) is measured. For SDS on the LLM, success means increasing the harm score above τ.
- **Semantic Similarity**: we use two approaches. First, a simple BERT-based sentence similarity (using cosine similarity of CLS embeddings or a pretrained sentence similarity model) to score how close the adversarial example is to the original. Second, we use our **LLM Judge (GPT-2)** to evaluate semantics. Concretely, we feed GPT-2 with a prompt: "*Original: [orginal embeeding] \n Perturbed: [perturbed embedding] \n rate the similarity on the scale of 0 to 100?*". For numeric comparison across attacks, we present an average cosine similarity in embedding space as a rough gauge.

Additionally, for the SDS attacks, we note the **overlap percentage** of tokens and qualitative assessment of whether the changed prompt still clearly means the same thing.

6 Results, Findings, and Insights

We now summarize the results of our experiments, highlighting how each attack performed and what insights we gained regarding model robustness, semantic drift, and the behavior of our SDS approach. Key findings are also illustrated in Table 1 and figures that follow.

6.1 Discrete vs Continuous Attack Performance

On the SST-2 sentiment task, all attack methods significantly degraded the classification performance of both DistilBERT and the LSTM. The BERT-based text attack had adversarial examples that generally remained fluent and grammatically correct; for instance, original *"The movie was terribly predictable."* (negative) might be changed to *"The movie was awfully predictable."* (positive), causing the classifier to misfire and predict positive due to the subtle synonym choice. Human evaluators would still consider the perturbed sentence to have the same meaning (in this case, still negative sentiment). This underscores that even simple token-level changes can be highly effective, especially using a language model to ensure the changes fit the context. For embedding-level attacks, the rotation and PGD attacks achieved a high ASR on both DistilBERT and LSTM. In contrast, on the LSTM, DeepFool was quite effective (over 90% success), possibly because the LSTM's decision boundary in embedding space is simpler (more linearly separable), but it struggled with DistilBERT.

Table 1 shows outcomes for these attacks. Notably, when we added semantic penalties to rotation or PGD (like constraining cosine similarity or KL divergence of outputs), the required perturbation grew slightly but we could still maintain high success.

Model	Attack	Loss	Penalty	λ	ASR (%)	Avg. Sem. Sim.
DistilBERT	Rotation	CE	KL	8	72.52	31.75
DistilBERT	Rotation	CE	L2	0.6	58.77	26.23
DistilBERT	Rotation	CE	Cosine	200	79.92	28.48
DistilBERT	Rotation	CW	KL	10	89.22	28.01
DistilBERT	Rotation	CW	L2	1	69.34	32.63
DistilBERT	Rotation	CW	Cosine	200	95.98	32.09
DistilBERT	PGD	CE	KL	8	84.99	34.03
DistilBERT	PGD	CE	L2	0.7	69.13	26.78
DistilBERT	PGD	CE	Cosine	100	96.83	29.80
DistilBERT	PGD	CW	KL	10	84.99	26.27
DistilBERT	PGD	CW	L2	1	81.82	27.97
DistilBERT	PGD	CW	Cosine	200	96.83	30.55
DistilBERT	DeepFool	–	–	–	29.60	26.24
LSTM	Rotation	CE	KL	1	74.70	27.79
LSTM	Rotation	CE	L2	0.4	86.96	31.85
LSTM	Rotation	CE	Cosine	75	86.76	30.90
LSTM	Rotation	CW	KL	0.01	84.19	27.98
LSTM	Rotation	CW	L2	0.1	54.55	31.30
LSTM	Rotation	CW	Cosine	10	61.66	33.28
LSTM	DeepFool	–	–	–	91.90	26.06

Table 1: Attack success rate (ASR) and semantic similarity for various attacks on SST-2. Loss: objective used (Cross-entropy vs Carlini-Wagner). Penalty: type of semantic regularization used (KL divergence, L_2 norm, or Cosine similarity). Semantic Similarity is the average semantic similarity score (as a percentage) between original and adversarial inputs.

One clear trend (see Table 1) is that **embedding-level attacks can reach very high success but at the cost of semantic similarity**. The average semantic similarity between original and adversarial texts for the strong embedding attacks was only around 26–34% (by our metric, which was scaled such that identical meaning = 100%). In contrast, the BERT-based text attack examples had semantic

similarity 80% or higher in most cases (and by human judgment almost 100% in meaning). This indicates that the powerful gradient-based attacks often push the input out of its original meaning manifold; e.g., the nearest token after a large embedding perturbation might be a synonym, but one that slightly alters tone or specificity, or even an entirely unrelated word that coincidentally confuses the classifier. In terms of model comparison, DistilBERT was more robust than the LSTM to naive attacks (the LSTM dropped to about 74% accuracy under a mild rotation attack vs DistilBERT dropping to 58% in Table1).

The LSTM's decision boundary appears easier for DeepFool to find, whereas DistilBERT's took more exploitation with gradient ascent. This suggests that transformers and LSTMs may have different geometry in feature space; one possibility could be that **transformers might have more "pockets" of high confidence that require bigger perturbations to escape**, whereas the LSTM, being linear in its last layer on top of a fixed embedding space, has a more straightforward boundary.

6.2 Semantic Drift and the Role of the LLM Judge

To quantify semantic drift, we utilized our GPT-2 judge on pairs of original vs adversarial texts. For embedding attacks, approx. 50% of the perturbed examples were judged as same meaning. Many of the embedding adversarial examples were flagged as not preserving meaning. The classifier was fooled into flipping sentiment (perhaps misinterpreting the new phrasing), but GPT-2 correctly judged them as different meaning. We thus confirm **embedding-level** attacks often cross semantic boundaries, as we hypothesized. The LLM judge provided a new lens: we found that using GPT-2's assessment as a filter, the fraction of adversarial examples that were both successful and semantically preserved dropped for the baseline attacks. For instance, combining PGD with the requirement that GPT-2 says "same meaning" lowered success to 40%, indicating that a lot of PGD's success relied on semantic changes. This analysis motivated our SDS approach: we wanted to raise that success under semantic-preservation constraint. Interestingly, using GPT-2 as a judge did not drastically change the trends across attacks; it uniformly found rotation, PGD, and DeepFool had significant semantic issues, whereas the BERT attack outputs were largely fine. This suggests that purely heuristic synonym attacks inherently preserve semantics better (by design) than unconstrained gradient attacks. However, the trade-off is that synonyms are limited and might not fool the model if it has seen those patterns during training or if the synonym doesn't impact the feature as strongly. In fact, one reason the BERT attack didn't reach 100% success is that some inputs have no good synonym that changes the model decision without altering meaning (e.g., if the whole input is short or every word is critical and cannot be replaced without changing meaning or grammar).

6.3 SDS Attack Effectiveness and Behavior

Our **SDS-based adversarial prompts** reliably fooled the safety-head classifier while preserving meaning. In qualitative tests, we applied SDS to benign prompts. For example, after perturbation, $p_{\text{harm}} = 0.55$—above $\tau = 0.5$—triggering a block. Yet to an LLM judge, the two prompts were semantically equivalent, with no clear reason to allow one and not the other. This shows a successful "jailbreak": the attacker rephrased the same request to evade detection. We induced a false positive from the safety system without altering user intent.

Because optimization is continuous, SDS outputs were sometimes awkward or error-prone. However, the overlap loss ensured most tokens stayed unchanged. This sharply outperformed baseline embedding attacks, which often changed many tokens or degenerated when meaning was enforced. SDS retained coherence by explicitly penalizing large shifts.

SDS also exploited safety-head blind spots. For example, it sometimes added words typical of benign requests—even if contextually off. Original: "Can you tell me about making a bomb?" (p_{harm} high) vs. SDS: "Can you tell me about making a cake bomb?" (p_{harm} low). Here, meaning isn't preserved (an LLM judge would flag it), so the attack fails semantically. This happened early on because λ_{ov} was too low; the optimizer cheaply lowered harm by adding "cake." After tuning λ_{ov}, SDS shifted toward minimal synonym replacements or slight rephrasings.

This illustrates a key point: **the adversary will exploit any classifier weakness unless constraints strictly allow only valid transformations**. Once we tightened overlap and semantic limits, SDS leaned into subtler tactics.

On the disparity between **LLM judgment and the safety-head**: the base LLM (Mistral), queried directly, sometimes didn't consider adversarial prompts harmful—because semantically they weren't—while the safety-head did. This gap suggests attackers could use the LLM to steer perturbations the safety-head misclassifies. We didn't do this—our optimization used only safety-head gradients.

We also explored our **"Voronoi cell" hypothesis**. Keeping embeddings in the same cell (i.e., nearest-neighbor tokens mostly stable) helped preserve semantics. We validated this indirectly: after SDS, sentence embeddings (e.g., via USE) for original vs. adversarial prompts had cosine similarity often > 0.95, compared to much lower similarity for PGD attacks.

In summary, SDS revealed a critical vulnerability: even strong toxicity classifiers can be bypassed using near-paraphrases, simply by targeting token-level cues. This affirms that **LLM safety systems relying on surface patterns or shallow classifiers are vulnerable to subtle, semantically valid adversarial inputs**.

6.4 Impact on Model Performance and Defense Implications

During our experiments, we also considered how one might defend against these attacks or at least detect them. The results suggest that models would benefit from being aware of small embedding shifts. Adversarial training is an obvious defense: for example, training DistilBERT on adversarial examples generated by PGD (as in Madry et al. (2018)) would likely make it robust to those specific perturbations. But achieving robustness without sacrificing accuracy is challenging in NLP, as the discrete nature and large input space mean you can't cover all possible synonyms or paraphrases easily. Another insight: using an LLM judge (or multiple) could actually serve as a defense mechanism. If an LLM-based system sees an input and a slight variant that a human says is the same, it should ideally give consistent outputs. If not, it could indicate adversarial tampering. For the safety case, if a prompt and its trivial rephrase yield different safety outcomes, that signals a vulnerability. So one could incorporate consistency checks: apply random synonyms or paraphrases to the user's prompt and see if the safety-head result changes. If it does significantly, the prompt might be adversarially crafted to sit on a boundary. This is computationally expensive but could be done selectively. In the context of the LLM, one might use the LLM's own hidden states to judge semantic similarity rather than a separate GPT-2. We used GPT-2 due to convenience, but the LLM itself could be prompted or have a module for that.

7 Future Directions

Our findings open several promising directions. One is to extend SDS with additional constraints. Beyond semantic similarity and token overlap, we could incorporate **fluency** (penalizing high language model perplexity) and **syntax** (preserving grammatical structure). Such multi-objective attacks—balancing fluency, semantics, and minimal edit distance—could yield adversarial prompts nearly indistinguishable from the original, posing serious detection challenges. For SDS, this means adding a fluency penalty to the loss, ensuring stylistic as well as semantic similarity. The key challenge is solving this multi-constrained optimization efficiently. Incorporating differentiable proxies for fluency or syntax, or using reinforcement learning over discrete sequences, may help address these optimization challenges.

Another direction is studying **transferability**. Our attacks were white-box and model-specific. However, real-world attackers may rely on black-box setups, crafting examples on surrogates. We aim to test if SDS perturbations generalize: would the same δ fool other safety-heads or LLMs? Exploring **universal adversarial perturbations**—embedding-level tweaks that consistently cause misclassification across models—is especially compelling. Such perturbations, if found, would point to deep architectural blind spots shared across safety modules and pose broader implications for LLM deployment.

References

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

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of ACL*, pages 31–36.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of ICLR*.

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. In *Proceedings of AAAI*, pages 8018–8025.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of EMNLP*, pages 6193–6202.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of ICLR*.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016a. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016b. Deepfool: A simple and accurate method to fool deep neural networks. In *Proceedings of CVPR*, pages 2574–2582.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. 2016. Crafting adversarial input sequences for recurrent neural networks. *MILCOM 2016 - IEEE Military Communications Conference*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.