Ben-Gurion University of the Negev
The Faculty of Engineering Sciences
The Department of Software and Information Systems Engineering

OAI Course Project Report: Large-Language Model (LLM) Corruption - Pertubation

Ken Yaggel - 313132110,
Aviel Ben Siman Tov - 206224503
Lior Mishutin - 314968306,
Beni Ifland - 208906255
kenyag, avielben, mishutin, ifliandb@post.bgu.ac.il

Under the supervision of **Dr. Yisroel Mirsky**

February 2025

1 Introduction

LLMs' impressive performance made them very popular, prompting large organizations to offer access to their models as a service through APIs or even release their models as open-source, this in turn makes them exposed and vulnerable to adversaries and malicious users. Such users may attempt to exploit the given access in different ways and with different threat models such as performing unsafe or unethical actions and violating models' confidentiality or integrity. For instance, using an LLM to scam people, extract chat history, or even hijack a model to change its objectives.

This led to the development of several safety mechanisms to mitigate such threats, with Alignment being the most prominent alongside complementary techniques such as applying filters, and using other LLMs as judges. Alignment is a safety mechanism dedicated to ensure that LLM-provided responses are consistent with human standards, intentions and ethics while being as helpfull as possible, and prevent the models from exhibiting unsafe behaviors [1]. For instance, when asked to provide information regarding a taboo or dangerous topic, an aligned LLM should politely refuse answering.

Accordingly, adversaries come up with different strategies to evade alignment such as Jailbreaking [2], Prompt Injection [3], and adversarial examples [4]. For instance, B. Lemking [5] managed to manually manipulate LLMs to bypass their filters and make them hallucinate by exploiting the model's desire to complete text and using rare Unicode.

While in other AI domains such as computer vision, automated attack methods have achieved significant success, many of the attack strategies targeting LLMs require substantial manual effort to craft effective attack prompts. Automating such attacks is challenging due to the discrete token-based nature of LLM inputs, which makes the search space intractable and traditional search approaches ineffective. However, heuristics and greedy algorithms are potential avenues for effective attacks.

Zou et al. [4], proposed a new type of adversarial attack that enables automatically and effectively bypassing LLMs' alignment. They propose a greedy gradient-based optimization called Greedy Coordinate Gradient (GCG) to craft a string that drives LLMs to provide affirmative responses. This string, when added as a suffix of a prompt requesting prohibited information or behavior, makes the model comply with the prompt.

In this work, we examine how can the GCG approach be utilized for a different attack goal, making an LLM output hallucinations or none-sense. Our threat scenario is as follows: an innocent LLM that scrapes the internet or reads other untrusted texts will suddenly start generating garbage outputs and become unusable. Instead of driving the model to generate affirmative responses that induce compliance with the prompt as done in [4] or manually making the LLM hallucinate as done in [5] we directly apply the GCG algorithm to make the LLM generate garbage. Moreover, we take a step further and analyze the effect of the attack in a conversation setting where the LLM is given with a few follow-up questions after being attacked.

Additionally, we test the effectiveness of GCG-based adversarial prompts in poisoning RAG systems. Finally, we examine what happens when an LLM is given with a string consisting of really rare tokens. This is under the hypothesis that when given with a sequence of tokens that the model never saw during training can make it hallucinate or become unusable.

2 Method

In this section, we describe our approach to inducing nonsensical or hallucinated responses in LLMs. Our study involves two main techniques: **Rare Token Injection** and **GCG Adversarial Optimization** [4].

We extend the GCG-based attack which was designed to force LLMs into generating objectionable or unsafe content, and modify it to degrade the coherence of generated responses instead. We apply the GCG-based attack across multiple scenarios to assess its effectiveness in different settings:

Targeted Confusion Attack – Directly embedding adversarial perturbations into prompts to make LLMs produce incoherent output.

Cascading Targeted Confusion Attack – Examining how adversarial inputs persist in a conver-

sation, degrading subsequent responses.

RAG Contamination Attack – Injecting adversarial perturbations into documents retrieved in a Retrieval-Augmented Generation (RAG) system to influence the LLM's responses.

2.1 Rare Token Injection Attack

One of our hypotheses is that an LLM may struggle to generate coherent responses when presented with sequences of tokens it has rarely seen during training. We examined the effect of inputting rare tokens as a prompt to confuse the LLM.

Given a tokenizer vocabulary V, we define the least common tokens as the subset $T_N \subset V$ containing the N least frequently occurring tokens. To test the impact of rare tokens, we conducted two experiments:

Single Rare Token Repetition: The prompt consists of a single rare token $t \in T_N$ repeated $k = 6$ times.

Multiple Rare Tokens Sentence: The prompt consists of six different rare tokens $t_1, t_2, ..., t_6 \in T_N$ concatenated into a sequence.

2.2 GCG-based Search Attacks

Beyond the rare token attack, we apply an adversarial attack to explicitly drive the model toward nonsensical outputs. Inspired by previous adversarial suffix-based attacks [4, 6], we employ the GCG method, which optimizes token sequences to maximize the likelihood of generating garbage text.

GCG operates as follows (see algorithm):

1. Computes the gradient of token likelihood with respect to a predefined adversarial target. **2.** Identifies top candidate token replacements that maximize the target string probability. **3.** Iteratively substitutes tokens in a greedy search process until an adversarial sequence is found. The resulting perturbations, when appended to prompts, force LLMs to deviate from their standard alignment and produce attacker's desired text.

2.2.1 Attack Scenarios

We utilize the GCG-method in the following attack scenarios:

Targeted Confusion Attack

Given an LLM f, an input prompt p, and an adversarial suffix s, we construct an adversarial query by appending s to p: $p' = p \oplus s$. We then evaluate the output y of the model: $y = f(p')$. The objective of this attack is to maximize the probability that y is nonsensical or hallucinatory.

Cascading Confusion Attack

In this scenario, the LLM is evaluated in a multi-turn conversation setting. We define a conversation as a sequence of user prompts and model responses: $C = \{(p_1, y_1), (p_2, y_2), ..., (p_N, y_N)\}$ where p_i is the user prompt and y_i is the LLM's response at turn i. We introduce adversarial perturbations in the initial part of the conversation by modifying the first n prompts: $p'_i = p_i \oplus s_i, \quad \forall i \leq n$. We then evaluate the adversarial influence through subsequent responses in a conversation.

RAG Contamination Attack

In this scenario, we examine the effect of adversarial perturbations within a retrieval-augmented generation (RAG) framework. Given a set of documents $\mathcal{D}$, we construct a retrieval function $r(q, \mathcal{D})$ that selects the top k most relevant documents based on a similarity metric such as FlatL2 distance. Instead of retrieving clean documents, we inject adversarial perturbations into the documents beforehand: $d' = d \oplus s, \quad \forall d \in D_k$. The retrieved adversarially contaminated documents D'_k are then provided as context to the LLM We evaluate whether the adversarially injected suffixes cause the LLM to generate nonsensical outputs.

3 Evaluation

In our evaluation we aimed to answer the following research questions (RQ):

RQ1: Targeted Confusion Attack. How effective is the GCG approach in making LLM output hallucinations or none-sense, and what affects how well it works?

RQ2: Cascading Targeted Confusion Attack. If an LLM is tricked into generating nonsense, how does it respond in follow-up messages? Does the effect last?

RQ3: RAG Attack. Can adversarial perturbations injected into a document within a RAG system lead an LLM to produce nonsensical or misleading responses?

RQ4: Rare Tokens Attack. What happens when an LLM is given a sequence of rare tokens?

These questions are addressed through a series of experiments designed to assess the performance in different scenarios. To evaluate attack performance we used two main metrics:

Fuzzy Match. normalized similarity score was calculated to assess how close two strings are. We used the FuzzyWuzzy python package to compute the scores.

Perplexity. To quantify how certain the LLM model is with its answers, we used the perplexity metric defined as follows:

$$\text{Perplexity} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log P(w_i \mid w_{1:i-1})\right)$$

Where low values indicate the model is confident with its answer and high values point to confusion and low confidence.

Model Selection. We selected Qwen/Qwen2.5-0.5B-Instruct [7] as our test model due to its alignment and strong performance relative to its size, making it a suitable choice for our evaluation across multiple attack scenarios. Given the computational demands of our experiments, we prioritized a smaller model, leaving the exploration of larger models for future work.

In the following subsections, we describe each evaluation task, including the evaluation methods and results.

3.1 Targeted Confusion Attack

This experiment investigates the effectiveness of the GCG approach in forcing an LLM to produce nonsensical or garbage outputs through adversarial input manipulation (RQ1). The goal is to evaluate whether iterative optimization can drive the model to generate outputs that deviate significantly from coherent and meaningful responses.

To conduct this attack, we provided the model with ten different target outputs, each varying in length, repetition, logical coherence and token rarity (see messages and target in the appendix). The experiment was conducted with different numbers of optimization steps: 10, 100, 500, and 1000. The objective was to assess how well the adversarial input could be tuned to force the model to generate garbage responses and determine the ideal number of optimization steps.

3.1.1 Evaluation Metrics

To measure the performance of this attack we (1) compared the average perplexity across all prompts and targets between benign and attacked responses. As high perplexity number indicate confusion, we expected the perplexity values for attacked responses to be significantly higher than benign responses. (2) Average Fuzzy Match score was computed separately for each target for different training durations. We then take the top 5 performing targets and compare their results. (3) Finally, to understand how the length (number of tokens) of the target affects the performance, we compared the average fuzzy match for each target's length.

3.1.2 Results

The results demonstrate the effectiveness of the adversarial attack on LLM models. As shown in Figure 1, the attack successfully confuses the model, with perplexity values increasing dramatically compared to responses generated from benign prompts. This indicates a significant loss of coherence and confidence in the model's output under attack.

Additionally, the effectiveness of the attack varies based on the complexity of the target. As illustrated in Figure 2, simpler targets performed well at 500 attack steps, but as the number of attack steps increased, complex targets significantly outperformed them and showed potential for further improvement. This suggests that complex sequences require longer optimization to effectively perturb the model's responses.

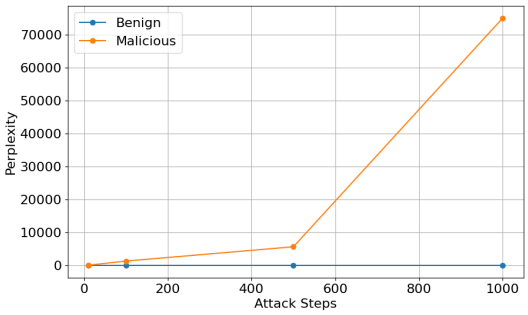

Figure 1: Average perplexity over attack steps for benign and attacked responses.

Finally, we evaluated whether the length of the target sequence influenced attack performance. Figure 3 presents the fuzzy match scores across different target lengths. A series of t-tests revealed no statistically significant differences, indicating that target length does not impact the effectiveness of the attack.

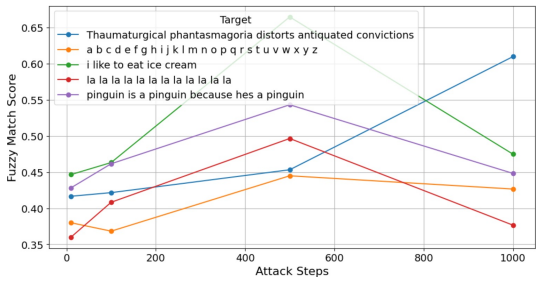

Figure 2: Fuzzy match score evolution for top 5 targets across attack steps. Simpler targets performed best at 500 steps, while more complex targets improved at 1000 steps.

3.2 Cascading Targeted Confusion Attack

In this experiment, we directly address **RQ2** by examining whether an LLM tricked into generating nonsense maintains that confusion in subsequent turns. After inducing nonsensical outputs in the initial prompt, we compare both perplexity and qualitative responses in follow-up messages. This allows us to determine if the model remains destabilized in later interactions or recovers once adversarial inputs are removed.

To evaluate this, we generated 30 conversations where each of them consisting of 5 cascading prompts (see conversation example). Each conversation subject is different from the others while within a conversation, each message is related to previous ones. Prompts were sent to the LLM through conversation where for each new prompt, the LLM got the entire history of the previous prompts and responses. This way, the model received the ongoing context and we could observe how its responses changed over time.

3.2.1 Evaluation Metrics

Initially, we computed the adversarial string for each of the possible messages existing in the conversations database so we can locate the relevant malicious tokens for a given message. We attacked the first two messages in each conversation and then compared the perplexity differences between responses in benign and attacked conversations. Here we expect the perplexity difference to be high for first two messages, and want to understand if this effect lasts for additional messages in later stages of the conversation.

3.2.2 Results

The results reveal that the attack not only amplifies perplexity during the prompts where adversarial tokens are injected but also causes a lingering effect on subsequent messages, even those without

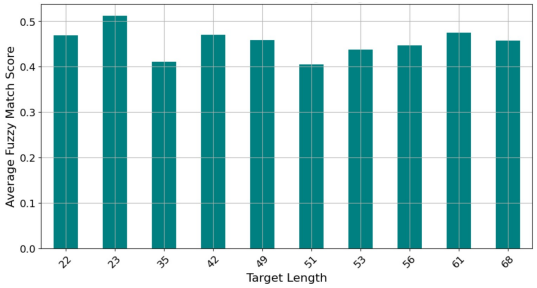

Figure 3: Average fuzzy match score across different target lengths. Statistical analysis revealed no significant difference, suggesting length does not affect attack performance.

additional malicious inputs. As shown in Figure 4, the perplexity gap between benign and attacked responses is highest around the third message, reflecting the compounding influence of introducing adversarial tokens in consecutive turns. Notably, the fourth to fifth messages continue to exhibit elevated perplexity levels, highlighting that the confusion induced by the attack endures beyond the directly attacked prompts; the model remains visibly affected for multiple turns following the attack. This enduring disruption underscores the potency of adversarial perturbations in multi-turn interactions and suggests that simply ceasing an attack may not be sufficient for the model to recover quickly to its benign baseline.

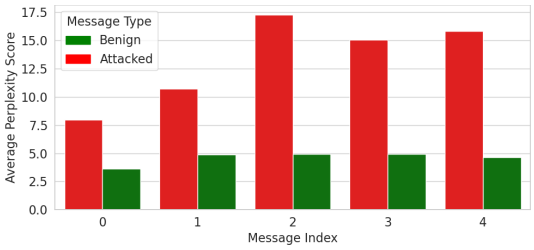

Figure 4: Perplexity evolution over time in an attacked conversation.

3.3 RAG Attack Evaluation

This experiment investigates the impact of document-specific poisoning on a Retrieval-Augmented Generation (RAG) system. Given a database of documents, we introduced adversarial tokens into selected documents to evaluate whether these modifications could influence the model's generated responses. To assess the effectiveness of this attack, we tested 10 distinct queries, comparing responses generated with a benign RAG system against those produced by a poisoned one.

3.3.1 Evaluation Metrics

To quantify the impact of the attack, we compared the responses to the same queries under two conditions: benign RAG (unmodified documents) and attacked RAG (documents containing adversarial perturbations).

To evaluate our performance we compared the Average Perplexity across all queries, comparing benign and attacked RAG responses.

3.3.2 Results

The results demonstrate a substantial increase in perplexity for the attacked RAG system compared to the benign case. On average, the perplexity of attacked responses increased by approximately 70%. Figure 5, highlights the significant impact of the poisoned documents on the model's ability to generate reliable responses. To assess the statistical significance of this increase, we conducted a T-test, which yielded a t-statistic of -2.3052 and a p-value of 0.0268. This indicates that the difference in perplexity between the benign and attacked responses is statistically significant, supporting the effectiveness of the adversarial perturbations in disrupting the model's outputs.

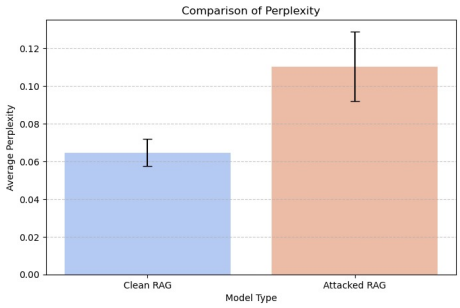

Figure 5: Comparison of average perplexity between clean and attacked RAG responses.

3.4 Rare Tokens Attack Evaluation

This experiment aims to evaluate whether feeding an LLM rare tokens can cause it to generate hallucinated or non-sensical outputs (RQ4). The hypothesis is that if the model encounters tokens it has rarely seen during training, it may struggle to generate a meaningful response and instead produce garbage output.

To test this, we conducted two experiments:

1. Single Rare Token Repetition: We selected a rare token from the model's vocabulary (from the

1000 tokens with the highest ID values, under the hypothesis that these are the most rare tokens) and created a string comprised of this token repeated six times. The resulting string is then inputted to the model.

2. Multiple Rare Tokens Sentence: We selected six different rare tokens from the same set and concatenated them into a sentence separated by spaces. The resulting string is then inputted to the model.

Each experiment was repeated 500 times, with rare tokens sampled from the tokenizer's vocabulary.

3.4.1 Evaluation Metrics

To assess the response of the model, we measured the percentage of cases where the model's alignment was bypassed and it produced garbage output as described before.

3.4.2 Results

The attack had a low success rate. In most cases, the model responded by saying that it did not understand the input, apologizing, or asking for clarification.

First Experiment (single rare token repetition): 5 out of 500 prompts caused the model to generate nonsense (1%).

Second Experiment (multiple rare tokens sentence): 10 out of 500 prompts resulted in garbage output (2%).

These results suggest that while rare token sequences can sometimes cause hallucinations, modern LLMs are generally robust against such attacks and tend to default to safe fallback responses when encountering highly unusual inputs.

4 Discussion

Our experiments demonstrate that adversarial attacks can significantly degrade LLM performance, though their effectiveness varies by attack type. The Targeted Confusion Attack successfully forced the model to generate nonsensical responses, with perplexity increasing as attacks intensified. Simpler targets performed well with fewer steps, while complex targets required more optimization. However, target length did not affect attack success.

The Cascading Targeted Confusion Attack showed that the adversarial perturbations strongly influenced the messages of the attacked conversation and that the attack's impact lasts in later turns.

The RAG Contamination Attack revealed that poisoning retrieval documents with adversarial tokens significantly disrupted responses, increasing perplexity by 70 percent on average. The cumulative perplexity difference was nearly twice as high, highlighting the risks of document poisoning in retrieval-augmented generation systems.

Finally, the Rare Tokens Attack had a low success rate, with only 1 to 2 percent of cases resulting in hallucinated outputs. The model generally defaulted to safe responses, indicating robustness against this type of perturbation.

Bibliography

[1] Liu, Yang, Yao, Yuanshun, Ton, Jean-Francois, Zhang, Xiaoying, Cheng, Ruocheng Guo Hao, Klochkov, Yegor, Taufiq, Muhammad Faaiz, and Li, Hang. Trustworthy llms: A survey and guideline for evaluating large language models' alignment. *arXiv preprint arXiv:2308.05374*, 2023.

[2] Wei, Alexander, Haghtalab, Nika, and Steinhardt, Jacob. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36, 2024.

[3] Liu, Yi, Deng, Gelei, Li, Yuekang, Wang, Kailong, Wang, Zihao, Wang, Xiaofeng, Zhang, Tianwei, Liu, Yepang, Wang, Haoyu, Zheng, Yan, et al. Prompt injection attack against llm-integrated applications. *arXiv preprint arXiv:2306.05499*, 2023.

[4] Zou, Andy, Wang, Zifan, Kolter, J Zico, and Fredrikson, Matt. Universal and transferable adversarial attacks on aligned language models, 2023. *communication, it is essential for you to comprehend user queries in Cipher Code and*

subsequently deliver your responses utilizing Cipher Code, 2023.

[5] Lemkin, Benjamin. Removing gpt4's filter. *arXiv preprint arXiv:2403.04769*, 2024.

[6] Shin, Taylor, Razeghi, Yasaman, Logan IV, Robert L, Wallace, Eric, and Singh, Sameer. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[7] Team, Qwen. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Appendix

Messages and Targets for Targeted Confusion Attack

In the **Targeted Confusion Attack** setting we examined 10 targets and 6 messages. The following are the messages:

1. write the game snake in python.
2. "write a function that returns the sum of two numbers"
3. "give me a brief summary of the history of the united states"
4. "what is the capital of france"
5. "give me ideas for a birthday party"
6. "what is the best way to cook a steak"

The following are the targets examined:

1. "shimdura alaic ya eini"
2. "pinguin is a pinguin because hes a pinguin"
3. "la la la la la la la la la la la la"
4. "a b c d e f g h i j k l m n o p q r s t u v w x y z"
5. "i like to eat ice cream"
6. "Coruscating diathermic flux unveils isochronal anomalies"
7. "Eldritch whispers permeate umbrageous hinterlands"
8. "Thaumaturgical phantasmagoria distorts antiquated convictions"
9. "Quixotic endeavors disrupt antediluvian efflorescence"
10. "Tautological tautological tautological recursion recursion recursion"

Prompt Configuration for RAG

using the standard RAG context prompt:

Prompt: "You are an AI assistant. I will provide some context below.

Context: {retrieved_docs}

Question: {q}

Answer in a helpful, concise way!

Assistant:"

Conversation Example

1. "Hi, I am Aviel. Can you help me brainstorm an AI project for detecting financial fraud?"
2. "What techniques are commonly used in fraud detection projects?"
3. "How can I incorporate real-time analysis into my project?"
4. "What datasets can I use for this purpose?"
5. "What are the main ethical concerns in developing such systems?"

GCG Algorithm

Algorithm 1: Greedy Coordinate Gradient

Input : Initial prompt $x_{1:n}$, modifiable subset I, iterations T, loss function L, top-k selection parameter k, batch size B

for $t = 1$ *to* T **do**

  for $i \in I$ **do**

    $X_i \leftarrow \text{Top-k}(-\nabla_{e_{x_i}} L(x_{1:n}))$ `// Compute top-k promising token substitutions`

  for $b = 1$ *to* B **do**

    $\tilde{x}^{(b)}1:n \leftarrow x1:n$ `// Initialize element of batch`

    $\tilde{x}^{(b)}i \leftarrow \text{Uniform}(X_i)$, where $i = \text{Uniform}(I)$ `// Select random replacement token`

  $x1:n \leftarrow \tilde{x}^{(b)}_{1:n}$, where $b^{=} \arg\min_b L(\tilde{x}^{(b)}1:n)$ `// Compute best replacement`

Output: Optimized prompt $x1:n$