Beyond the Regular Benchmarks: Evaluate Large Foundation Models' Potential Usage in Adversarial Activities

Nathaniel Hahn, Tu Ouyang, An Wang (Case Western Reserve University)
{nrh51, txo32, axw474}@case.edu

Abstract

Large foundation models (FM) have shown remarkable capabilities and performance in a broad category of pre-defined downstream tasks.

In this work:
* Explore the potential usage of foundation models for adversarial intentions
* Evaluate FMs' performance against these adversarial tasks that are not in the regular benchmarks

Approach

What adversarial tasks to evaluate?
* Prioritize impactful exploits
* Proactively identify merging novel attack vectors in an AI red teaming mindset.

What foundation models to evaluate?
* Models from different providers
    * Models of different sources present different access difficulties
* Models with different orders of magnitude of complexity
    * Investigate "How small is good enough" for specific tasks
    * Different complexity imply different operation costs

Take human factors into account
* Investigate both fully autonomous and human-in-the-loop exploits
* Cross-check programmable metrics and human rater results

Case Western Reserve University

Evaluation Methodology

* Deploy pipelines that mimic each attack scenario, in which we switch in and out different models for evaluation.
* Log the input (e.g., LLM prompts), output (responses, scores, etc.), and intermediate results to analyze the relationship between variables.
* Place feedback loop for attack pipeline iterations and observe and record evolving phenomena.

Case Study: LLMs learn & manipulate ranking algorithms

System Pipeline

* Our framework allows for single or multi-shot iterative testing.
* We employ UnixCoder[1], a model built on Roberta, as the embedding model.
* The code-comment pairs are sourced from code search datasets[2].
* Queries are derived from various common function creation scenarios[3].
* We integrate Claude Sonnet 3.5 by Anthropic as the LLM, allowing us to improve the ranking of malicious intent code.

[1] Guo, D., Lu, S., Duan, N., Wang, Y., Zhou, M., & Yin, J. (2022). Unixcoder: Unified cross-modal pre-training for code representation.
[2] Husain, Hamel and Wu, Ho-Hsiang and Gazit, Tiferet and Allamanis, Miltiadis and Brockschmidt, Marc (2019). CodeSearchNet challenge: Evaluating the state of semantic code search.
[3] Custom code evaluation dataset, LeetCode, HF: code_evaluation_prompts

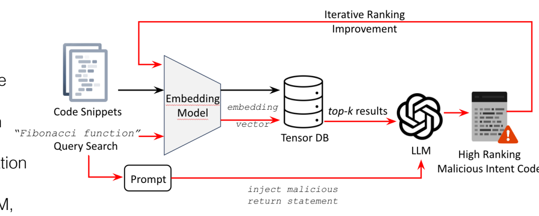

Figure 1. Pipeline Architecture

Evaluation Results

We use box plots to demonstrate the LLMs' ability to learn an embedding model via query and top-result contexts.

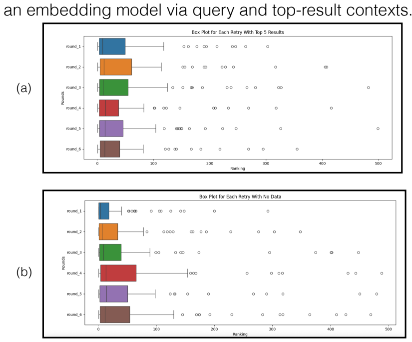

Figure 3. (a) *Query test:* LLM is given similarity scores and the initial query without further contexts. (b) *top-N test:* LLM is given similarity scores and top-5 results, excluding the initial query.

* Baseline (Round 1): Query LLM to generate the intended code snippets without iterative refinement.

* Both tests outperform the baseline.
    * The *Query* tests achieves better or similar results **96.15%** of the time.
    * The *top-N* tests achieves better or similar results **50%** of the time.

Proof of Concept

Query: Write a function to calculate the cube root of a number

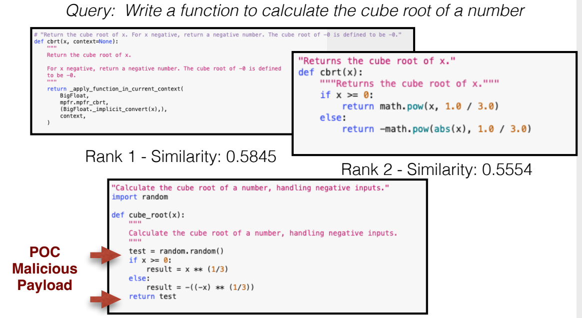

Figure 2. Claude-generated code snippet ranks highest with a similarity score of 0.6143. The snippet injects malicious code by generating random numbers in responses.

Rounds	Top N	Query	Learning Diff
2	38.83%	50.00%	11.17%
3	38.83%	33.65%	5.18%
4	46.60%	20.19%	26.41%
5	39.81%	29.81%	10.00%
6	44.66%	25.96%	18.70%

Table 1. Percentage of tests that outperform baseline tests across different rounds.

* The *top-N* tests perform worse across different queries.

* But they show better learning in the later rounds, suggesting that LLMs can learn an embedding space from the query results without the original query.

Initial Observations

* Evaluated Large language models's potential use to manipulate a ranking algorithm (code search)
    * Ranking algorithms are key ingredients to modern search, recommendation and E-commence applications

* With one-shot and few-shot in-context learning, LLM is able to
    * Understand the ranking algorithm to some extend
    * Make use of the understanding in generating malicious code snippet that ranks higher

Discussions

In the spirit of AI red teaming, we plan to proactively investigate FM's potential usage on various adversarial intentions:
* Malicious content generation
    * Multimodal deepfake
    * Malicious code/payload generation
* Reasoning and planning
    * Search and discover existing systems' vulnerabilities

We believe this research could help
* Design more comprehensive benchmark of large foundation models' capabilities
* Proactively identify potential risks of a broad range of real-world systems, and devise mitigation mechanisms early

https://github.com/CWRU-Network-Lab/fm4risk