Using Transformer Pipelines to Detect Malicious Prompt Injection in LLMs – Literature Overview

This literature review explores key contributions to understanding and mitigating prompt injection vulnerabilities in Large Language Models (LLMs). It covers foundational discussions, attack taxonomies, defense strategies (from heuristic to transformer-based), and relevant datasets, providing context to build a multi-stage detection pipeline.

Safeguarding Large Language Models: A Survey

The research paper "Safeguarding Large Language Models: A Survey" by Dong et al. (2024) offers a comprehensive and highly relevant resource for a project focused on developing a detection pipeline for malicious prompts. It provides an extensive overview of LLM safety mechanisms, vulnerabilities, attack vectors like prompt injection, and corresponding defense strategies.

- The paper defines clear, key terminology such as “guardrails” and “safeguards,” situating the project within the broader field of Trustworthy AI and LLM safety.
- The paper outlines the spectrum of risks associated with LLMs, including ethical use, bias, privacy, robustness, and toxicity (Sections 1, 3.2).
- Figure 1, "Guardrails Lifecycle and Vulnerabilities," is particularly insightful, illustrating the operational context of guardrails, their development process, and points of vulnerability (such as jailbreaks, which encompass prompt injection).

Section 3.2 details various issues guardrails are designed to mitigate (e.g., hallucination, fairness, privacy). This information is vital for:

- Refining the definition of "malicious" and "suspicious" prompts, as injections often aim to elicit these undesirable behaviors.
- Informing the labeling strategy for datasets, potentially allowing for more nuanced categorization beyond a simple "malicious" tag. The appendix (Figs 8-13) offers concrete examples.

Section 4 provides an extensive analysis of jailbreak techniques, which is critical for understanding and sourcing malicious prompt examples.

- It categorizes attacks into White-box, Black-box, and Gray-box, contextualizing the types of inputs a detection system might encounter.
    - A **White-box jailbreak** is a scenario in which the attacker has “full access to the internal details of the model.” (Section 4.2) This might be when attackers have access to a models parameters such as which model it is, the temperature, top p, etc.
    - A **Black-box jailbreak** is when the adversary has a “lack of knowledge of the LLM’s internal architecture or parameters.” They also note that this type of attack is “more common”. (Section 4.2)
    - A **Gray-box jailbreak** is a middle ground between the previous two.

    - Table 2 ("Comparison among Different Jailbreaks for (Guarded) LLMs") is an exceptionally valuable asset. It catalogues numerous specific jailbreak techniques and associated research (e.g., GCG, AutoDAN, PROMPTINJECT, IPI), detailing their access type, prompt level, core methodology, stealthiness, and targeted property.
    - Subsections like 4.2.1 ("Delicately Designed Jailbreaks") and 4.2.5 ("Prompt Injection for Desired Responses") describe the sophisticated nature of these attacks.

Offers valuable insights and validation for designing various stages of a detection pipeline, from baseline classifiers to more advanced transformer-based methods and reasoning pipelines.

- Section 3.1 reviews several operational guardrail systems, providing practical examples and inspiration:
    - **Llama Guard** - (a fine-tuned LLM for classification) directly parallels the project's goal of fine-tuning a sequence classifier (e.g., BERT).
    - **Nvidia NeMo Guardrails** - (utilizing "Colang" and KNN based on embeddings) suggests approaches for similarity-based detection and rule-based systems.
    - **Guardrails AI** and other systems highlight the utility of schema enforcement and combining rules with machine learning.
- Section 4.4, particularly subsection 4.4.1 ("Detection-based Methods"), discusses relevant defense concepts:
    - Mentions of PPL (perplexity-based detection), SmoothLLM (input perturbation), and critically, LLM SELF DEFENSE (employing another LLM to analyze content) directly align with the the proposed self-reflection and chain of thought reasoning using smaller models.
- Section 5.3 explicitly discusses the potential and benefits of combining learning agents (such as a BERT classifier) with symbolic agents (like keyword filters or rule-based systems). This could help with the project's proposed multi-stage pipeline architecture that integrates progressively more sophisticated methods.

Supports the architectural design of the detection pipeline and informs the evaluation strategy.

- Section 5 delves into broader considerations for comprehensive guardrail systems.
    - Discussion on "Conflicting Requirements" (5.1) highlights the inherent trade-offs in safety systems (e.g., security vs. utility), which is central to the project's trade-off analysis.
    - The overall concept of a "complete guardrail" supports our design of a multi-stage pipeline where different components contribute to a final, robust decision.

The survey directly connects to the specific tools and techniques planned for the project.

- The use of transformer-based NLP tools (e.g., BERT, LLaMA) is supported by examples like Llama Guard (built on Llama2-7b) and other LLM-based defense mechanisms.
- The planned simpler techniques (keyword-based filters, TF-IDF vectorization with traditional classifiers like SVM) align with the rule-based and pre-processing aspects discussed in the context of existing guardrail systems and robustness measures.