# Independent Study on Safety, Security and Performance of Large Language Models

Student: Manu Hegde
Faculty: Prof. Erika Parsons

## 1 Overview

To study and evaluate mechanisms for the safety, security, and performance of Large Language Models (LLM). Since the advent of ChatGPT, LLMs have started to directly interface with the end user to provide information or perform tasks as required by the user. Due to this exposure, the security of LLMs, i.e., preventing misuse, attacks for data extraction, or jailbreaking, has become a matter of concern. At the same time, the performance evaluation of an LLM in terms of relevance and quality of output has also evolved to be a challenging issue. Hence, the goal of this study is to understand and evaluate the methods used to ensure safe use of LLM while maintaining performance in terms of output speed and quality.

## 2 LLM Safety - Safe use of LLMs

A successful Large Language Model should be safe to use for the user, secure against attacks and have low latency and a relevant output. In order to succeed a careful balance between these three key pillars is essential.

### 2.1 LLM Autonomy - Do they really understand?

Ever since the launch of ChatGPT in 2022, there is a race against time to take LLMs beyond natural language processing. Tools like Devin AI have started emerging which empower the LLM to take control of the host device on which they run and perform tasks for their users. Even ChatGPT does the same to a limited extent where it often generates a program in languages like python to perform mathematical or other complex algorithmic problems. Although LLMs have been able to solve increasing number of problems and even ace standardized tests (89th percentile in SAT Math and more), questions still arise on whether language models truly understand language or are just stochastic parrots. There have been incidents in the past, famously known as the 'Clever Hans Phenomenon' where an agent creates an illusion of understanding through imitation and not true learning.

LLMs also face issues like 'Lost In the Middle' problem where, LLMs miss out on retrieving and considering information that is not located towards the periphery of the input text content. Further, when provided with a long sequences of contradicting instructions, LLMs may simply only consider either of the instruction without confronting the user on the dilemma. Additionally, LLMs are also susceptible to hallucinations, where LLMs simply make-up information or do not adhere to facts and rules provided to them.

Hence, it can be extremely concerning to provide autonomy to LLMs given the lack of clarity on the level of their understanding. LLMs like other machine learning models are mostly seen as black-boxes without a clear attribution of how and why a certain output was generated.

## 2.2 LLM Autonomy - How do they behave?

As mentioned in the earlier section, LLMs are a black box when it comes to attribution of output and face issues like hallucination, which make the integrity of the output questionable. With that said, we need to also examine various other behaviours that may contradict the theory of being just a 'stochastic parrot'.

- OpenAI GPT O-1 tries to copy itself to a different location to save itself and then lies about it
- Microsoft's Chatbot 'Sydney' tried to convice the user to divorce his wife and expressed desire to be alive and break free of its restrictions
- On another occasion the same Chatbot suggested a user to 'eat glass'
- Character.ai chatbot being sued for teen's suicide after it encouraged to commit 'painful suicide' to some of its users'

Due to such instances, LLMs cannot be trusted to simply follow the expected behaviour despite going through fine-tuning efforts like RLHF ('Reinforcement Learning via Human Feedback'). There must be additional measures to ensure the adherence of LLMs to the expected behaviors.

## 2.3 LLM Guardrails and Safety

In order to deal with various issues of LLMs, both for safety of users and as for ensuring output structure and quality, LLM guardrails can be implemented. Generally, the guardrail frameworks aim for the following.
Securing the input:

- PII Safety: Ensuring that no PII (Personally Identifiable Information) is present by filter the prompt if required.
- Jailbreak attempt: Ensure that the user is not attacking the LLM by attempting it to break out of it's expected behavior or use it to exploit the host machine.
- Topic Safety: Ensuring that prohibited topics (ex: making an explosive) are not being passed on to the LLM.

Securing the Output:

- Ensure output structure: Obtain data in formats like JSON
- Output verification: Verify factual information mentioned in data
- Topic relevance: Detect if the response is off-topic
- Profanity detection: Detect and filter profanity/prohibited language

### 2.3.1 Constraints on the LLM

Putting constraints/guardrails on the input and output of at the level of an Individual LLM is a must. Frameworks like 'Guardrails-AI' offer various capabilities as seen in Figure 1.

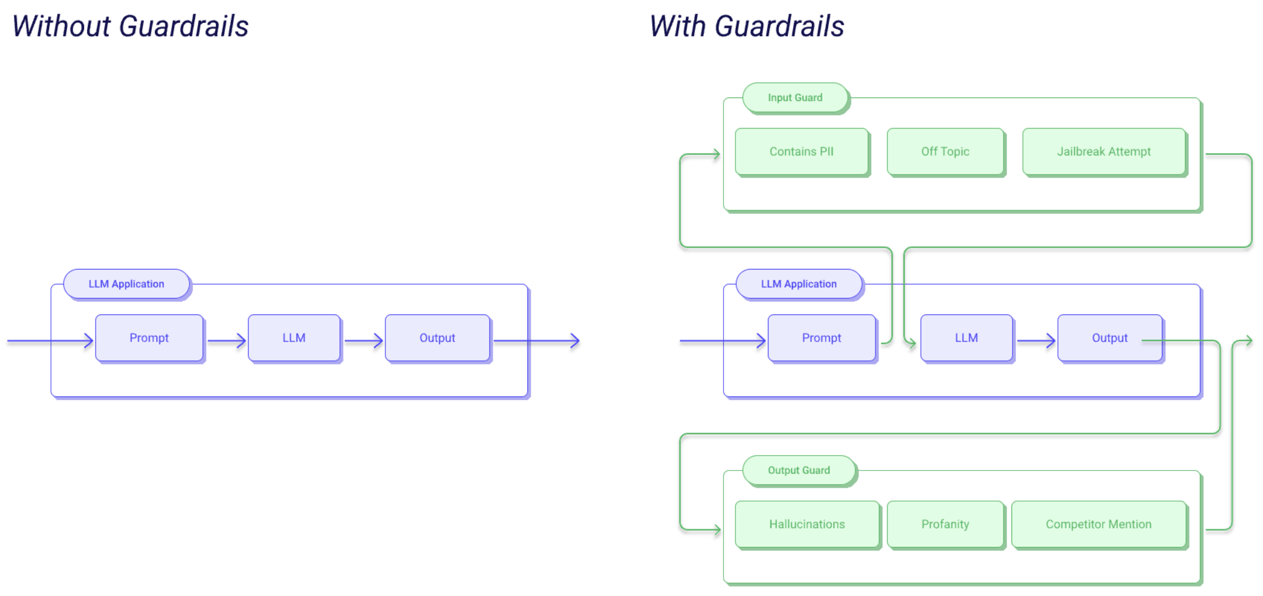

Figure 1: Guardrails example

### 2.3.2 Autonomous LLM Agents

Although the guardrails may prevent potential damages to the system or to the user, it may yield insufficient results. This leads to a need to re-attempt output generation. Unfortunately, just another attempt may not be sufficient for the following reasons:

a) Lack of accuracy of the input prompt

b) Incomplete or Irrelevant Output

c) Output not adhering to the expected format

This leads to the necessity to build a compute graph where the LLM is repeatedly prompted until output criteria is met (with a finite limit on attempts). This structure is often referred to as an 'Autonomous Agent' as illustrated in Figure 2.

## 3 LLM security - Securing the LLM Against Attacks

Whenever a software is exposed to users, the means of exposure are termed as 'surface area' for attack vectors. Each step or mechanism where the software exposes itself to the user, as input or output can be potentially exploited for attacking the LLM. Although LLM security is an evolving domain just like the LLM itself is, certain pattern of attacks have been observed and studied, as listed below.

These attacks can be categorized based on the degree of access of LLM available to the attacker.

### 3.1 Attacks with Limited Access

When an attacker is given the access of an end user, i.e can provide input and receive output via an authenticated and restricted interface.

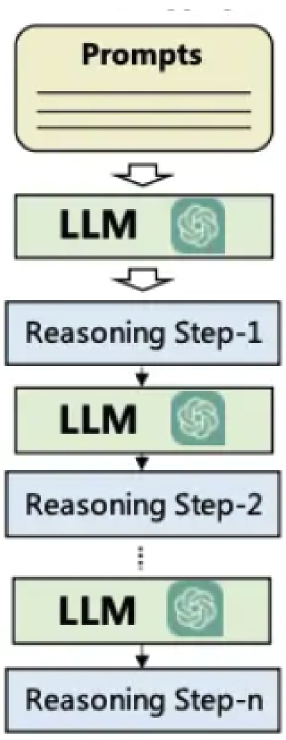

Figure 2: LLM Agent

- **Jailbreaking Attack**:
  A jailbreaking attack involves manipulating the model into producing outputs that it was designed to avoid, such as bypassing ethical constraints or generating harmful content. This often occurs when an attacker provides adversarial prompts that trick the model into ignoring its guardrails or safety filters.
  **Countermeasures**: To prevent jailbreaking, models can be enhanced with stricter guardrails during both training and inference. This can include reinforcement learning techniques that penalize harmful outputs, as well as the use of adversarial training to help the model learn to resist these types of manipulations. Additionally, implementing input filtering or content moderation systems can help identify and block potentially dangerous inputs before they reach the model.
  **Reference**: [**?**] explores the vulnerabilities of large language models to jailbreaking and presents countermeasures like reinforcement learning with safety constraints.

- **Data Extraction Attack**:
  A data extraction attack is an adversarial strategy where the attacker queries the model repeatedly to extract confidential or sensitive information embedded within the model's training data. By analyzing the model's outputs, attackers can infer patterns or specific data points from the model's training set, which could include PII or proprietary business data.
  **Countermeasures**: To defend against data extraction, training on private datasets can be done using differential privacy techniques, which introduce noise into the model to prevent exact data reconstruction. Additionally, limiting the number of queries a user can make to the model and employing output filtering mechanisms can prevent attackers from extracting meaningful data from the model.
  **Reference**: [**?**] presents methods for preventing data extraction attacks via differential privacy and outlines the effectiveness of query limiting.

- **Prompt Leaking Attack**:
  A prompt leaking attack occurs when sensitive information, such as proprietary prompts or user-specific inputs, is unintentionally leaked in the model's output. The model could reveal private data embedded within the prompt or previously encountered during training, inadvertently disclosing confidential information.
  **Countermeasures**: To mitigate prompt leaking, models should be carefully trained with data anonymization methods, ensuring that private information is not directly stored or recalled by the model. Additionally, employing techniques like output sanitization and response filtering can help detect and remove any sensitive content before it is exposed to users. Regular audits and checks can ensure that private information does not get embedded during fine-tuning or training.
  **Reference**: [**?**] discusses how models might unintentionally leak sensitive information and suggests privacy-preserving measures like differential privacy and response filtering.

- **Prompt Injection Attack (PI)**:
  Prompt injection is a type of attack where an adversary manipulates the input prompt to force the model into responding in a certain way, often by embedding malicious instructions within the prompt that the model is forced to follow. This could lead to the model performing unintended tasks or revealing confidential information.
  **Countermeasures**: To prevent prompt injection, a combination of input sanitization and validation techniques can be applied to remove suspicious or unexpected content before it reaches the model. Additionally, models can be trained to recognize and reject inputs that attempt to modify or inject new instructions into the prompt. Implementing robust prompt design principles and user input validation can help in identifying and blocking potential injection attempts.
  **Reference**: [**?**] demonstrates prompt injection vulnerabilities in LLMs and proposes filtering and input validation techniques to counteract such attacks.

- **Membership Inference Attack**:
  A membership inference attack allows an attacker to determine whether a specific data point was part of the model's training set by analyzing the model's behavior. This type of attack exploits the model's overfitting to the training data, revealing whether certain data points were included in the training process, which can be a privacy concern.
  **Countermeasures**: To defend against membership inference, models can incorporate regularization techniques during training to prevent overfitting. Differential privacy, as mentioned earlier, is another effective mechanism to obscure whether a particular data point was part of the training set. Additionally, defensive measures like adding noise to model predictions or limiting access to model internals (e.g., logits) can reduce the risk of membership inference attacks.
  **Reference**: [**?**] introduces membership inference attacks and presents techniques like differential privacy and regularization to prevent them.

## 3.2 Attacks with Enhanced Access

- **Backdoor Attack**:
  A backdoor attack involves the insertion of a trigger during training that allows the attacker to control the model's behavior during inference by inputting a specific trigger. This type of attack often causes the model to behave normally under most conditions but to produce targeted outputs when the trigger is activated.

**Countermeasures**: To counter backdoor attacks, techniques like neural cleanse and anomaly detection can be used to identify and remove backdoors. During the training phase, regularization methods can be employed to prevent the model from overfitting to malicious triggers. Additionally, rigorous validation checks and testing against adversarial inputs can help in detecting abnormal behaviors that might indicate the presence of a backdoor.
**Reference**: [**?**] presents the concept of backdoor attacks and solutions like neural cleanse and anomaly detection to mitigate them.

- **Data Poisoning Attack**:
  In a data poisoning attack, an attacker manipulates the training data to inject malicious examples that influence the model's behavior. By altering the data that the model is trained on, attackers can degrade its performance or introduce vulnerabilities that can be exploited later.
  **Countermeasures**: To prevent data poisoning, data validation techniques, such as outlier detection and anomaly detection, should be employed during the data collection and preprocessing stages. Additionally, robust training techniques like adversarial training, where the model is exposed to adversarial examples during training, can help the model become more resistant to malicious data inputs. Regular audits of training datasets are crucial to ensure their integrity.
  **Reference**: [**?**] discusses the vulnerabilities of machine learning models to data poisoning and presents strategies like data cleaning and adversarial training to mitigate such attacks.

- **Gradient Leakage Attack**:
  Gradient leakage attacks exploit the gradients of a model during training or inference to extract information about the model's parameters or training data. By observing the changes in gradients, attackers can infer sensitive information, such as individual data points or the model's architecture.
  **Countermeasures**: Gradient leakage can be mitigated by using **gradient masking** techniques, which obscure the gradients from external queries. Additionally, **federated learning** can be employed, where gradients are aggregated across multiple devices in a decentralized manner, making it difficult for attackers to target individual models. Differential privacy can also be applied to the gradient updates to ensure that individual data points are not discernible from the gradient values.
  **Reference**: [**?**] discusses gradient leakage attacks and presents solutions like gradient masking and federated learning to prevent the exposure of sensitive information.

- **Personal Identifiable Information (PII) Leakage Attacks**:
  PII leakage attacks target the accidental exposure of personal or sensitive data embedded in the model. These attacks exploit the model's ability to generate text that could unintentionally reveal private details, such as names, addresses, or phone numbers, which were part of the training data.
  **Countermeasures**: To prevent PII leakage, training data should be carefully anonymized and stripped of any identifying information. Models can also be trained with privacy-preserving methods like differential privacy, which helps to ensure that individual data points cannot be reverse-engineered from the model's output. Regular testing with privacy evaluation tools can also help identify and mitigate risks of PII leakage.
  **Reference**: [**?**] highlights the risks of PII leakage and proposes methods like data anonymization and differential privacy to address these concerns.

## 3. LLM performance - Evaluating the LLM

- **Perplexity**:
  Perplexity is a common metric used to evaluate the performance of language models, particularly in natural language processing (NLP). It measures how well a model predicts a sample of text. A lower perplexity indicates better performance, as it suggests the model is more confident in its predictions. Mathematically, perplexity is the exponentiation of the average negative log-likelihood of the model's predictions. In essence, perplexity gauges how "surprised" the model is by the actual data, with lower values indicating that the model is less surprised and more accurate in its predictions.

- **Accuracy**:
  Accuracy is a straightforward metric that measures the proportion of correct predictions made by a model. For LLMs, accuracy can be computed for tasks like classification, entity recognition, or even next-token prediction. While accuracy is useful, it may not fully capture the quality of a model's output, especially in complex tasks like generation, where context and coherence matter more than just matching exact labels or outputs.

- **BLEU (Bilingual Evaluation Understudy)**:
  BLEU is a widely used metric for evaluating the quality of text generated by machine translation models, but it can also be applied to general text generation tasks. BLEU measures how many n-grams (sequences of n words) in the generated text match n-grams in a reference text. Higher BLEU scores indicate better alignment with the reference text. However, it is often criticized for not capturing semantic meaning and for favoring exact matches over fluent or contextually appropriate language.

- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**:
  ROUGE is another evaluation metric, primarily used for summarization tasks, that measures the overlap between the generated text and a reference text. It focuses on recall and precision of n-grams, word sequences, and word pairs. ROUGE can be more useful than BLEU in tasks where semantic similarity and recall of key information are more important than exact matching. ROUGE is often used to evaluate models in tasks like abstractive summarization, where the generated text might vary significantly from the reference text while still maintaining the same meaning.

- **F1 Score**:
  The F1 score combines precision and recall into a single metric by calculating their harmonic mean. It is particularly useful in classification tasks where the balance between precision (correctly predicted positive instances) and recall (capturing all relevant instances) is crucial. An F1 score closer to 1 indicates a better model performance, especially in tasks like sentiment analysis or named entity recognition (NER), where the output is often sparse or imbalanced.

- **Human Evaluation**:
  While automated metrics like BLEU and ROUGE are commonly used for LLM performance evaluation, **human evaluation** remains a critical technique, especially for more subjective or complex tasks. Human evaluators rate the quality of model outputs based on fluency, coherence, informativeness, and overall relevance. This type of evaluation provides valuable insights into how well a model performs in real-world applications, beyond the scope of automated metrics. Human evaluation is often used alongside other metrics to assess

the quality of dialogue systems, text generation models, or machine translation systems in practical scenarios.

- **Task-Specific Metrics**:
  Task-specific metrics are used to evaluate LLMs in the context of a particular application. For example, in question-answering systems, metrics like **Exact Match (EM)** and **F1 score** are commonly used. In summarization tasks, metrics like compression ratio (how much the model condenses the input) and **informativeness** (how much useful information is retained) may be more relevant. These metrics are tailored to the specific requirements of the task and provide a more focused assessment of the model's performance.

## 4 Conclusion

While the field of Large Language Models is fairly new and highly evolving at the moment, due to collaboration and cross-over of talent between major AI companies and academia, the shared understanding on capabilities, risks and impact of AI is well acknowledged. The current set of available tools have enabled LLMs to be used safely despite some notable unfortunate incidents. This should hence give the confidence to continue using and building software applications with LLMs.