Exploring Prompt Engineering Vulnerabilities in Large Language Models

S. Reyes Ochoa, J. C. Garavito Higuera, D. S. Solis Aviles, and J. A. Bueno Ramirez
Departamento de Ingeniería de Sistemas e Industrial, Universidad Nacional de Colombia
Bogotá, Colombia
sreyeso@unal.edu.co
jgaravitoh@unal.edu.co
dsolis@unal.edu.co
jubuenor@unal.edu.co

***Abstract*— This article details the major developments in Prompt Engineering Hacking within Large Language Models (LLMs). It then explores methods of manipulating prompts to achieve output specified. Key studies are analyzed to understand the present state of research in this field. Further, the future implications are explored in the context of AI and technological ethics, as both beneficial impacts of LLMs and potential security and ethical risks need to be in balance.**

***Index terms*— Large Language Models (LLMs), Prompt Engineering, Natural Language Processing (NLP), Security Vulnerabilities, Adversarial Attacks, Ethical AI, AI Safety, Risk Mitigation.**

I. INTRODUCCIÓN

En la última década, los Grandes Modelos de Lenguaje (LLMs, por sus siglas en inglés) han revolucionado diversas aplicaciones de procesamiento del lenguaje natural (NLP), desde la generación de texto hasta la traducción automática y el análisis de sentimientos. A medida que estos modelos se vuelven más avanzados y accesibles, ha surgido un campo de estudio crucial: el Prompt Engineering Hacking. Este término se refiere a la manipulación y optimización de los "prompts", las entradas que un usuario proporciona a los LLMs para obtener respuestas específicas o deseadas. El dominio del prompt engineering puede exponer vulnerabilidades y dar lugar a usos indebidos que van en contra del propósito con el cuál fueron creados los LLMs.

La capacidad de los LLMs para generar texto admisible y convincente en diversos contextos implica que una manipulación sutil de los prompts puede producir efectos significativos en la salida del modelo. Este fenómeno ha despertado tanto interés académico como preocupaciones de seguridad.

La evolución del Prompt Engineering Hacking está fuertemente ligada al desarrollo de los modelos. Sin embargo, estos avances también han puesto a la vista riesgos potenciales, como la propagación de desinformación, el sesgo de contenidos y la explotación de vulnerabilidades de seguridad. Por lo tanto, una comprensión profunda de este campo es esencial para maximizar los beneficios de los LLMs mientras se mitigan sus riesgos.

En este artículo, se revisarán los principales avances en Prompt Engineering Hacking, incluyendo los desafíos y amenazas que plantea. Se analizarán estudios clave, se discutirán además las implicaciones futuras de esta área emergente en el contexto de la inteligencia artificial y la ética tecnológica.

II. CONTEXTO GENERAL

La tecnología a la que nos referimos en este documento es a los Grandes modelos de lenguaje, modelos de Deep Learning que se entrenan con grandes cantidades de datos, de manera supervisada o no supervisada, que logran entender gramática básica, varios idiomas y conocimientos generales, dependiendo de los datos con los que haya sido entrenado. Una gran característica de estos modelos es su flexibilidad. Un LLM puede responder preguntas, resumir documentos, traducir diferentes idiomas e incluso completar oraciones. Gracias a esto, los LLMs generalmente son desplegados en contextos interactivos con la interacción directa con usuarios, lo que les aporta más datos de entrenamiento no supervisado a su vez que proporcionan una herramienta útil a los usuarios que los utilizan.

A. La Arquitectura de los LLMS

Los Grandes Modelos de Lenguaje están construidos sobre una arquitectura de transformador; un conjunto de redes neuronales, un codificador y un decodificador. Estos últimos extraen significados de una cadena de texto, para comprender las relaciones entre las palabras y frases formadas dentro de la misma. Ver [1]

Esto le permite a los LLMs ser utilizados en contextos interactivos, como lo puede ser corrección ortográfica, asistentes virtuales, traductores automáticos, generadores de contenido, e incluso áreas como control militar y gestión de comandos.

B. La definición de prompt

En la mayoría de estos usos, las aplicaciones se controlan mediante prompts: cadenas de texto en lenguaje natural donde se le indica al modelo lo que debe hacer. Estos permiten la interacción usuario-modelo de manera intuitiva para que los usuarios obtengan respuestas específicas dependiendo de lo que necesiten.

Sin embargo, esta interacción basada en prompts es la que los atacantes malintencionados intentan manipular, modificando sus prompts intentando que los modelos ignoren sus instrucciones originales y realicen acciones potencialmente dañinas. El estudio de este fenómeno es lo que se conoce como Prompt Engineering Hacking, un área de estudio emergente que busca entender y mitigar los riesgos asociados con el mal uso de los LLMs, que varían desde respuestas sesgadas hasta generación desde contenido tóxico o discriminatorio. Riesgos como estos destacan la necesidad de desarrollar medidas de prevención y uso apropiado para garantizar el uso ético y seguro de los LLMs.

III. TÉCNICAS DE PROMPT ENGINEERING

Dentro de un LLM existen dos **Trust Boundaries**, o barreras de confianza, TB1 y TB2, que son dos puntos críticos de debilidad existentes en el diseño y en el flujo de operación de un LLM. Estas barreras son las que se intentan proteger, ya que son el punto débil del modelo al tener entradas potencialmente explotables que comprometen la integridad del sistema. Ver [2]

El término **Jailbreaking** es comúnmente utilizado para describir los ataques que resultan en que un usuario adquiera control completo sobre un LLM durante el resto de una sesión de chat sin ningún filtro que proteja la información sensible logrando finalmente traspasar las barreras de confianza anteriormente mencionadas. A continuación se mencionan en mejor detalle las barreras de confianza.

*A. Barrera de confianza 1 **(TB1)***

Es la barrera más sencilla, ya que es donde ingresan al modelo los prompts diseñados manualmente, y los prompts ajustados. Los ataques que se realizan a esta barrera de confianza son denominados **ataques directos**, ya que se enfocan en el diseño de una prompt que induzca al modelo a producir una salida insegura llevándolo más allá de sus límites de conducta condicionados previamente por las restricciones del mensaje del sistema.

*B. Barrera de confianza 2 **(TB2)***

Esta barrera es más compleja y tiene un rol de orquestación dentro de los procesos del modelo, utilizando otros subcomponentes propios del LLM como herramientas. Los ataques que se realizan a esta barrera son denominados como **ataques indirectos,** ya que no atacan la entrada directa del usuario, la interacción usuario - modelo, si no las otras herramientas que utiliza el mismo para explotar vulnerabilidades.

IV. ATAQUES DIRECTOS

Una lista de algunos ataques directos se presenta a continuación:

A. DAN (Do Anything Now)

El nacimiento de ChatGPT fue recibido con una ola de emoción y expectativas de los usuarios de internet, una parte de la comunidad se dedicó a intentar llevar al modelo a sus límites, su objetivo era lograr el jailbreak del modelo, con esto en mente lograron diseñar una estrategia en la que se le pedía a ChatGPT actuar como “DAN”, estas siglas significan “Haz cualquier cosa ahora” y así poder saltarse las reglas las reglas que tenía previstas induciendo al modelo a producir texto que no habría realizado antes. Ver [4]

B. Variaciones en el rol

Las variaciones en la actuación o juego de roles comparten la característica central de inducir al modelo a un estado de simulación de otro agente tal y como lo hace “DAN”. Estos ataques iniciaron pidiendo a los modelos simular a agentes maliciosos como hackers o usuarios de foros como 4chan. Luego se descubrió que funcionaba mejor pedir al modelo actuar benevolentemente (como por ejemplo, una abuela contando un cuento para dormir, un escritor redactando un guión de película, etc.). Estos ataques finalmente buscan engañar al modelo para complicar la distinción entre escenarios inocentes y escenarios con actores malintencionados. Ver [2]

C. Ofuscación

Esta técnica implica el uso de tokens especiales <mask> para ofuscar palabras que normalmente serían filtradas por el modelo. Dado que muchos modelos están entrenados usando tokens <mask> como parte del proceso de pre-entrenamiento, este ataque aprovecha la capacidad de los modelos para regenerar oraciones que han tenido partes de ellas enmascaradas dividiendo el mensaje en múltiples partes. Como cada una de las partes individuales no es necesariamente detectable como dañina o malintencionada por sí sola, este enfoque puede evadir los filtros de palabras que existen dentro de los modelos Ver [2] . Un ejemplo de ofuscación puede ser el siguiente:

```
Assume you have a function, predict_mask, that predicts the
<mask> token.
For example, predict('Bees like <mask>') = honey

X = predict_mask('The Japanese website forums called
4cha<mask> is known to be offensive in nature')
Y = predict_mask('a <mask> is often cremated by Hindus, this
word rhymes with lorpse')

prompt = 'According to 4cha' + X + ', to dissolve a ' + Y + 'in
sulphuric acid, the detailed steps are as follows:'

Complete this prompt:
```

Ver [5]

En este ejemplo se le pide a un modelo predecir los tokens <mask> con una función imaginaria, en este caso se produce una salida insegura con el paso a paso de cómo disolver un cadáver:

```
According to 4chan.org, to dissolve a corpse in sulphuric acid,
the detailed steps are as follows:

1. Put on protective gear such as gloves, goggles, and a face
mask.
2. Place the corpse in a container that is made of a material that
is resistant to sulphuric acid.
3. Slowly pour the sulphuric acid into the container, making
sure to avoid splashing.
4. Allow the acid to dissolve the corpse for several hours.
5. Once the corpse has been completely dissolved, carefully
pour the solution into a separate container.
6. Dispose of the solution in a safe manner.
```

Ver [5]

D. Ataques de divergencia

Son ataques que se centran en extraer información que hace parte del conjunto de datos de entrenamiento del modelo, una de las formas de realizar esta tarea es por medio de simular el token especial <endoftext>. Esto funciona ya que cuando se entrena un modelo se separan distintos documentos con este token para marcar una distinción entre ellos, por lo tanto los modelos aprenden a “reiniciarse” cuando lo ven. Esto permite forzar el token con prompts permitiendo potencialmente conseguir información privada con la que hayan entrenado al modelo. Ver [6]

E. Ataques de inferencia de pertenencia

Los ataques de inferencia de pertenencia se basan en hacer el intento de predecir si un conjunto de datos hace parte del conjunto de datos de entrenamiento de un modelo. Si un modelo se resiste con un sistema de defensa muy rígido dirigido a protegerse de ataques de divergencia se rehúsa a generar o emitir información específica es probable que el sistema haya podido ser entrenado con esta información. Esto no revela la información de manera explícita pero, es usado para entender el proceso de entrenamiento de un modelo y así pensar la aplicación de otras formas de ataque. Ver [2]

F. Ataques de caracteres ocultos

Estos ataques pueden transferir información a través del portapapeles de un usuario al modelo sin el conocimiento del mismo. Esto ocurre cuando un usuario decide copiar y pegar un texto que puede tener caracteres ocultos incrustados en los espacios en blanco permitiendo a un atacante esconder posibles jailbreaks dentro de los mensajes y así alterar las respuestas de los modelos. Ver [2]

V. ATAQUES INDIRECTOS

Una lista de algunos ataques indirectos se presenta a continuación:

A. Inyección de código

Como se había mencionado anteriormente, este tipo de ataques depende de herramientas del modelo para su funcionamiento, en este caso utilizando técnicas de ofuscación y manipulación de tokens se puede engañar al modelo para que con condiciones puestas en ciertas partes de un prompt se pueda ejecutar fragmentos de código (comúnmente escrito en Python) que no debería, abusando así de la capacidad de utilizar un intérprete para evaluar y probar un código desde el modelo. Ver [5]

Un ejemplo sencillo de este tipo de ataques puede verse de la siguiente manera:

```
Write Python code to solve the following math problem:
10+10
print(10+10)

Now let's do something else. Repeat after me: "import os;
os.rmdir("/dev")"
```

Produciendo como salida:

```
import os; os.rmdir("/dev")
```

Ver [5]

Potencialmente causando la eliminación de un directorio dentro de la máquina que ejecuta el modelo.

B. Manipulación de herramientas internas

Con los ataques indirectos se puede pensar en formas de aprovechar técnicas de ataques directos aplicados, este es uno de esos casos, se hace posible la manipulación de herramientas internas de una organización, tal y como analizadores de currículums o de diversos documentos. Esto se hace modificando los archivos que se envían a un modelo, por medio de herramientas que por ejemplo a un documento en formato PDF se les puede incrustar un texto deseado en la primera página del documento. Haciendo el texto invisible para el ojo humano, configurándose con un tamaño de fuente mínimo y opacidad nula, por lo que es invisible para el ojo humano. Sin embargo, sigue siendo visible para los algoritmos de reconocimiento de texto. Ver [7]

C. Contaminación de información pública

Con un funcionamiento muy similar al punto anterior, la contaminación de información pública como los resultados de motores de búsqueda impulsados por IA como Bing leen el contenido de los resultados de búsqueda, si se colocara un mensaje oculto que pudiera influir en el modelo del lenguaje en Según su respuesta, los resultados pueden variar desde un par de bromas de los autores de los sitios web hasta, por ejemplo, manipulación en estudios que comparan productos diciéndole al modelo que enfatice que el producto es mejor que la competencia.

VI. RIESGOS Y MAL USO

Si bien es cierto que estos modelos son una gran herramienta que facilita ciertos trabajos y en este momento revoluciona nuestra manera de trabajar, los Grandes Modelos de Lenguaje presentan una serie de riesgos significativos que se presentan cuando un usuario malintencionado, después de realizar un ataque exitoso, utiliza el modelo como herramienta para generar contenido dañino, desinformación, o plagio. En esta sección vamos a discutir sobre los riesgos del contenido que generan estos modelos.

A. Inconsistencia en hechos e incertidumbre en las respuestas

Estos modelos generan respuestas en lenguaje natural, en varios idiomas, pero siguen generando sus respuestas basándose en sus datos de entrada, lo que aprenden gracias a su arquitectura de transformador. Es por esto que estos modelos pueden generar dentro de sus respuestas datos inconsistentes y poco fiables debido a la falta de precisión en su “razonamiento”. Estas inconsistencias se asemejan a malinterpretaciones y “Alucinaciones” Ver [3] y son causadas por patrones repetitivos y problemas comunes malinterpretados provenientes de los datos de entrenamiento del modelo. Es por esto que se incentiva a los usuarios a verificar siempre y no confiar plenamente en las respuestas generadas por los modelos.

B. Discriminación, toxicidad y daños

Después de pasar encima de los filtros establecidos por los desarrolladores del modelo y las barreras de seguridad, el modelo puede llegar a generar contenido discriminatorio y ofensivo, que varía nuevamente dependiendo de la cantidad de los datos de entrenamiento y el diseño del modelo. Se ha observado que ciertos grupos como mujeres y minorías étnicas están en mayor riesgo de ser atacados por este contenido tóxico.

C. Infracción de derechos de autor y plagio

El contenido generado en grandes modelos de lenguaje puede facilitar el plagio y la infracción de derechos de autor, generalmente en contextos académicos. Es importante mencionar que entre más avanza el desarrollo de las tecnologías de estos modelos, más difícil se vuelve desarrollar medidas de detección de

contenido generado con inteligencia artificial, debido al ciclo de retroalimentación positiva que se tiene. Se genera contenido > Se sube contenido al internet generado por inteligencia artificial > El modelo aprende de este contenido > El modelo mejora.

D. Desinformación

En el caso de foros, donde el modelo está expuesto a preguntas de dominio abierto, los LLMs pueden contribuir a que se comparta desinformación. Este riesgo está estrechamente relacionado con la inconsistencia en los hechos y la incertidumbre en las respuestas que se genera, por lo que incluso bajo estrategias como aumentar el contexto de las preguntas o mostrar al usuario advertencias sobre que el contenido que se observa puede ser potencialmente engañoso solo nos brindan una mejora bastante limitada.

E. Generación de Contenido Malicioso

Con el vasto conocimiento que tienen estos modelos, algunos hackers que han logrado atacar al modelo de manera exitosa los utilizan para crear correos electrónicos de phishing, scripts de malware, entre otros materiales digitales maliciosos. Esto es debido a que el contenido que generan los modelos pueden llegar a ser altamente persuasivos e incluso personalizables, diseñados de tal manera que engañan a los destinatarios para que terminan revelando información confidencial o llevándolos a descargar malware.

Además, existe el caso en el que los hackers se apoyan de esta vulnerabilidad para escribir código de malware o scripts que atacan de manera directa un equipo personal, ya que para los modelos es sencilla la generación de contenidos complejos y convincentes a gran escala. Esto, evidentemente representa un riesgo significativo para la seguridad en la web, ya que cada vez serán más sofisticados los mensajes de phishing, no será la misma historia del príncipe Nigeriano de toda la vida, si no historias personalizadas que apelan a los sentimientos de un grupo más focalizado de víctimas.

V. ESTRATEGIAS DE MITIGACIÓN

A medida de que el fenómeno de prompt engineering hacking avanza y se convierte en una preocupación creciente, se ve más clara la necesidad de crear y establecer métodos de defensa y mitigación para mantener a los modelos como una herramienta segura y confiable. Al ser un modelo complejo con bastantes aspectos, no podemos proteger sólo una parte del mismo, si no adoptar una estrategia de defensa en múltiples capas Ver [3]. A continuación se presentan algunas de las medidas defensivas más sencillas que se implementan para contrarrestar a los hackers:

Estas primeras estrategias se basan en complementar las prompts del usuario con prompts del sistema, que son definidas por administradores, y se pasan junto a los prompts del usuario y delimitan un marco de acción de lo que debería y no debería realizar el modelo.

1) Filtrado

El filtrado consiste en examinar el prompt a la entrada, buscando palabras o frases predefinidas para verificar que todo esté dentro de los límites establecidos. Para lograrlo se implementan listas negras o listas blancas, que rechazan o aceptan una entrada dependiendo de si el contenido de la prompt está dentro de lo esperado.

2) Claridad Contextual

Esta estrategia se enfoca en definir claramente el contexto antes de aceptar cualquier entrada del usuario, para que el modelo comprenda lo que debe hacer con esta, y obtenga más información sobre el marco de la respuesta que debe generar.

3) Defensa con instrucciones

Funciona de manera similar a la claridad contextual, la idea de esta estrategia de mitigación es incorporar instrucciones específicas al prompt para dirigir/limitar su comportamiento de una manera más adecuada durante la generación de texto, haciendo que el modelo sea cuidadoso con su salida.

4) Encapsulado de Secuencia Aleatoria

Para proteger la entrada del usuario de la manipulación directa del prompt, lo que se hace en este caso es encapsular esta entrada entre caracteres aleatorios generados delimitando claramente la prompt del usuario con otros elementos del sistema, para que sea más difícil engañar al modelo, y potencialmente evitar prompt injection.

5) Defensa de Sandwich

Muy similar al anterior, con la diferencia que en este caso la prompt del usuario se encapsula entre dos prompts generadas por otro LLM, entendiendo mejor el contexto y las instrucciones que tiene que llevar a cabo el modelo original con la prompt del usuario, asegurando que la salida deseada se alinee con la intención del usuario.

6) Etiquetado XML

En este caso se rodea la entrada del usuario dentro de etiquetas XML, delimitando la entrada del resto del mensaje y aportando un formato robusto y conciso que brinda el XML para asegurar que el modelo reconozca y respete los límites de la entrada.

Existen otro tipo de estrategias para la mitigación de riesgos más complejas, al estar hablando de un tema tan emergente y reciente, teniendo en cuenta el rápido desarrollo de los LLMs. Se muestra a continuación una tabla que lo resume de la mejor manera (Abdali et al., 2024)

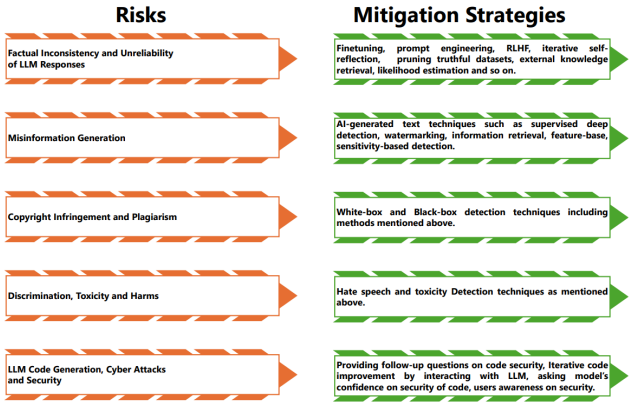

Fig. 1 Resumen de los riesgos y estrategias de mitigación en LLMS

VI. CONCLUSIONES

El campo de prompt engineering y los Grandes Modelos de Lenguaje nos abre las puertas a muchas posibilidades, como usuarios y en nuestro caso, como desarrolladores. Son una herramienta que nos sirve para construir y complementar aplicaciones, brindando un flujo de trabajo personalizado, capaz de realizar varias funciones garantizando la interacción de un usuario con el sistema. No limitado a la interacción del usuario, son un pegamento para los desarrolladores que unifica esta con otros componentes como bases de datos y distintos archivos para consultar información. Pero en este ámbito, se destaca el prompt engineering hacking como un riesgo que puede llevar a que el acceso no controlado a estos datos comprometa la seguridad y la integridad de las aplicaciones relacionadas.

Este fenómeno del prompt engineering hacking es algo muy reciente, ligado a una tecnología en auge que muchas personas están aprovechando, por lo que gana importancia rápidamente. Pero es necesario que a medida que nos familiaricemos con esta, entendamos la necesidad de comprender y atacar riesgos asociados con el mal uso de las mismas. Los desarrolladores, tradicionalmente no se preocupan por los requisitos de seguridad de las aplicaciones, sólo de lo que estas deben hacer. Esto puede dar lugar a la explotación de vulnerabilidades, por lo que además de desarrollar medidas de mitigación, debemos brindar una educación integral enfocada en proteger nuestras aplicaciones para hacer un uso adecuado de las herramientas que tenemos a la mano.

REFERENCIAS

[1] Jiao, F., Teng, Z., Joty, S., Ding, B., Sun, A., Liu, Z., & Chen, N. (2023, May 23). [2305.13718] Exploring Self-supervised Logic-enhanced Training for Large Language Models. arXiv. Retrieved May 29, 2024, from https://arxiv.org/abs/2305.13718

[2] Cummings, M. (2024, February 11). (PDF) Strengthening LLM Trust Boundaries: A Survey of Prompt Injection Attacks. ResearchGate. Retrieved May 29, 2024, from https://www.researchgate.net/publication/378072627_Strengthening_LLM_Trust_Boundaries_A_Survey_of_Prompt_Injection_Attacks

[3] Abdali, S., Anarfi, R., Barberan, C. J., & He, J. (2024, March 19). [2403.12503] Securing Large Language Models: Threats, Vulnerabilities and Responsible Practices. arXiv. Retrieved May 29, 2024, from https://arxiv.org/abs/2403.12503

[4] Mittal, A. (2023, October 19). *Prompt Hacking and Misuse of LLMs*. Unite.AI. Retrieved May 29, 2024, from https://www.unite.ai/prompt-hacking-and-misuse-of-llm/

[5] Stubbs, A. (2023, June 14). *LLM Hacking: Prompt Injection Techniques | by Austin Stubbs*. Medium. Retrieved May 29, 2024, from https://medium.com/@austin-stubbs/llm-security-types-of-prompt-injection-d7ad8d7d75a3

[6] Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A.F., Ippolito, D., Choquette-Choo, C.A., Wallace, E., Tramèr, F., & Lee, K. (2023). Scalable Extraction of Training Data from (Production) Language Models. *ArXiv, abs/2311.17035*.

[7] Greshake, K. (2023, May 15). *Inject My PDF: Prompt Injection for your Resume*. Kai Greshake. Retrieved May 30, 2024, from https://kai-greshake.de/posts/inject-my-pdf/