.UBAfiuba
FACULTAD DE INGENIERÍA

Aplicación de modelos extensos de lenguaje en la ambientación narrativa

Mario Gómez Alonso

Carrera de Especialización en Inteligencia Artificial

Director: Bach. Josselyn Sofía Ordóñez Olazábal

Jurados:

Jurado 1 (pertenencia)
Jurado 2 (pertenencia)
Jurado 3 (pertenencia)

Ciudad de Madrid, España, junio de 2025

Resumen

En la presente memoria se describe el diseño e implementación de un prototipo que emplea modelos extensos de lenguaje para la asistencia de los creadores en la generación de contenido narrativo. El trabajo se realizó para la aplicación de la empresa Critical Match en su desarrollo de una de sus ramas de ampliación de servicios para sus usuarios.

Para su implementación fueron imprescindibles los conocimientos adquiridos en la especialización tanto en su contenido teórico como en los procedimientos para la creación y programación de proyectos de inteligencia artificial, entre las que se destacan asignaturas como procesamiento de lenguaje natural y modelos extensos de lenguaje.

Agradecimientos

Esta sección es para agradecimientos personales y es totalmente **OPCIONAL**.

Índice general

Índice de figuras

Índice de tablas

Dedicado a... [OPCIONAL]

Capítulo 1

Introducción general

En este capítulo se presenta el contexto general y la motivación del trabajo realizado. A su vez, se realiza un análisis del estado del arte relacionado con las tecnologías que se utilizaron y se establecen los objetivos y el alcance definidos durante su realización.

1.1. El desafío de la creación narrativa

La narrativa[1] es un género literario que relata un conjunto de sucesos protagonizados por uno o más personajes y que son presentados a través de un narrador. Se trata de una forma de expresión cultural fundamental, presente en todas las culturas y épocas, utilizado para entretener y transmitir conocimiento. Una de sus características esenciales es la presencia de elementos ficticios, ya sea de manera parcial o total, con la única excepción del subgénero de crónicas que se limita a relatar hechos reales. La narrativa sigue siendo un elemento central de la cultura y está presente en múltiples formatos como la literatura, el cine, la televisión y otros medios tanto físicos como digitales.

La inclusión de elementos ficticios en la narrativa es un proceso creativo en el que el autor recurre a su imaginación para concebir componentes que, aunque inventados, resulten verosímiles y significativos. En este ejercicio mental intervienen múltiples factores, como la coherencia interna y la relación entre los distintos elementos que conforman la historia en el espacio y el tiempo. Estas variables se vuelven especialmente complejas en narrativas con una alta carga ficticia donde se inventan, además de los personajes y sucesos, contextos completos. Aquí es donde cobra especial relevancia la construcción de mundos[2], un recurso narrativo fundamental que permite al escritor diseñar entornos imaginarios detallados y capaces de sostener la lógica del relato.

El proceso de creación de mundos es tan complejo como la narrativa que busca sustentar y la intención del autor de sumergir al lector dentro de la historia. En la figura 1.1 se presenta una lista amplia, aunque no completa, de componentes fundamentales en la construcción de mundos y la manera en que estos se relacionan entre sí para dar cohesión al conjunto narrativo.

Históricamente los autores han recurrido a diversas herramientas para llevar a cabo la construcción de mundos, tales como esquemas, mapas, cronologías, fichas de personajes y la creación de lenguas artificiales. [1] Todos estos elementos

[1] Un ejemplo popular es el de J.R.R. Tolkien, quien creó mapas, genealogías y lenguas para su obra *El Señor de los Anillos* [3].

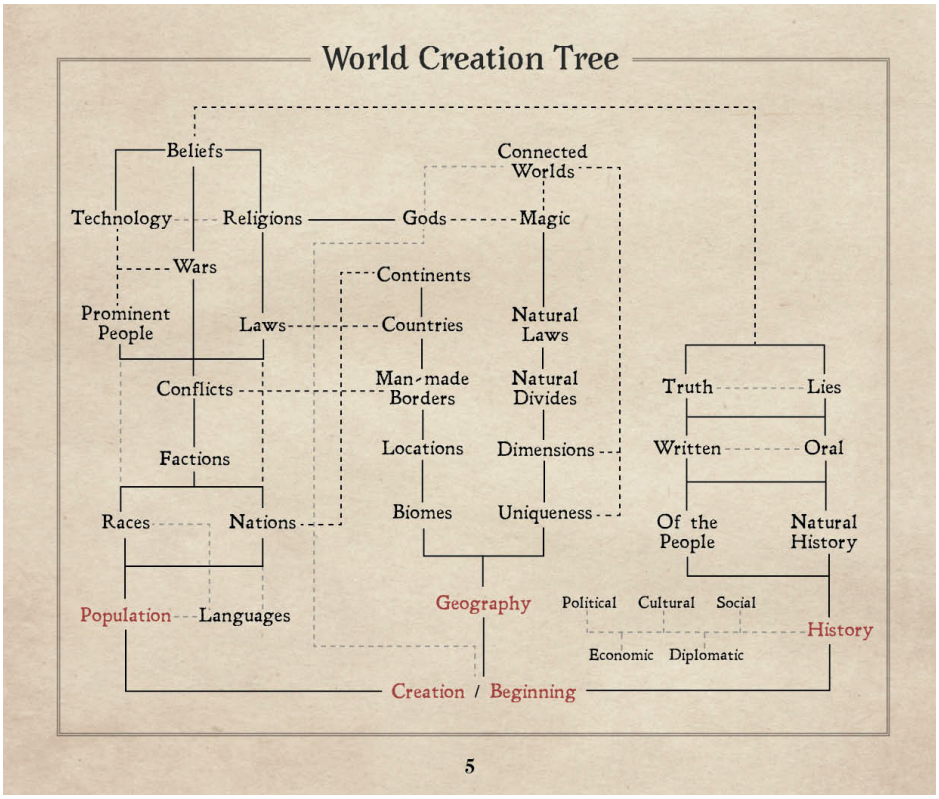

FIGURA 1.1. Esquema de elementos habituales en la construcción de mundos

han sido producto del trabajo creativo del autor, apoyado tanto en su imaginación como en la consulta de múltiples referencias. Este proceso ha requerido una considerable inversión de tiempo y, en muchas ocasiones, afectado por bloqueos creativos a la hora de articular y cohesionar todos los componentes ficticios de la narrativa.

En los últimos años la aparición de la inteligencia artificial generativa ha abierto nuevas posibilidades dentro de los entornos creativos. Aunque originalmente no fue concebida con fines narrativos, su capacidad para producir texto coherente la convierte en una herramienta con un alto potencial para incorporarse en la construcción de mundos ficticios. Sin pretender sustituir la creatividad del autor, estas tecnologías pueden ejercer un papel de apoyo al facilitar el desarrollo de contenidos y aportar ideas que estimulen la creación literaria.

1.2. Motivación

El trabajo se enfocó en la creación de entornos narrativos para juegos de rol y surgió de la colaboración con la empresa Critical Match[4] y su objetivo de ampliar las herramientas y servicios para los usuarios de su aplicación de móvil. La *app*[5] permite la creación y búsqueda de salas de juego que se ajusten a las necesidades y preferencias de cada usuario. En ese contexto, se busca mejorar la ambientación de las partidas proporcionando al autor de la narrativa a una serie de servicios para enriquecer el universo narrativo ya creado.

El trabajo aborda la necesidad de los usuarios de conseguir partidas más inmersivas y reducir el trabajo de ambientación del dueño de la sala de juego. Reconociendo que no todos los jugadores tienen experiencia con la creación narrativa, esta herramienta facilita la generación de una ambientación atractiva. De este modo se optimiza el tiempo de preparación del director de juego, permitiéndole centrarse en otros aspectos de la partida. Esto también beneficia a los más experimentados al expandir sus ideas con mayor rapidez y eficacia.

Los servicios implementados en el trabajo pretenden ofrecer una experiencia claramente diferenciada de las alternativas más populares en internet [2]. Para ello, se proporciona a los usuarios el acceso a consultas especializadas que solo requieren el envío de un fichero con la información narrativa, pudiendo combinarse adicionalmente con indicaciones que pueden escribirse en un cuadro de texto. Este enfoque no solo evita la incomodidad de interactuar con el modelo a través del navegador de internet del móvil, sino que ofrece respuestas precisas a sus necesidades, que requerirían interacciones complejas con las alternativas existentes.

Además, este desarrollo se complementa a herramientas existentes de creación de mundos [3]. Mientras que estas plataformas brindan la estructura y el marco para la creación de mundos, el trabajo ofrece la capacidad de generar dinámicamente contenido narrativo específico con la información ya existente. De este modo los autores a través de ambas herramientas pueden centrarse en la visión general y cohesión del mundo mientras que la inteligencia artificial los ayuda con ideas con las que expandir la ambientación, lo que fomenta un proceso cíclico creativo.

1.3. Estado del Arte

El avance de la inteligencia artificial generativa ha seguido un ritmo extraordinario desde la publicación del influyente artículo *Attention is All you need* [6] en 2017. La arquitectura de los transformadores revolucionó el procesamiento de lenguaje en las máquinas y permitió paralelizar los procesos a través del mecanismo de atención. Esto aumenta drásticamente la velocidad y eficacia en el proceso de aprendizaje de los modelos.

En el transcurso del año siguiente surgieron los primeros modelos extensos de lenguaje que adoptaron la arquitectura de los transformadores. Con el lanzamiento de GPT[7] por OpenAI y de BERT[8] por Google, ambos modelos se convirtieron en los pioneros y pilares fundamentales de los modelos extensos de lenguaje. Mientras BERT se especializó en la comprensión del contexto del texto de manera bidireccional, GPT fue desarrollado para la generación de texto.

Desde entonces la evolución de los modelos no se ha detenido. El número de parámetros ha crecido exponencialmente, lo que les ha otorgado una capacidad sin precedentes para comprender y producir texto coherente y fluido similar al humano. Además, este desarrollo ha sido transversal y está dotando a los modelos más recientes con la habilidad para generar no solo texto, sino también imágenes, audio e incluso video.

Estos avances están actualmente al alcance del público general a través de internet, con ejemplos destacados como ChatGPT[9], Gemini[10] o Meta AI[11].

[2] Algunos ejemplos son ChatGPT de OpenAI, Gemini de Google o MetaAI.

[3] Por ejemplo: WorldAnvil, Obsidian Portal o Legend Keeper.

ChatGPT fue el pionero y el que abarcó mayor popularidad, con una interfaz gráfica similar a la de un chat con la que el usuario podía interactuar de forma intuitiva y sencilla con el modelo. La utilización de estos modelos se masificó rápidamente y se implementaron variantes en multitud de sectores, principalmente en forma de asistentes virtuales.

En el ámbito creativo, el potencial de la inteligencia artificial generativa es tan amplio como discutido. Dependiendo de su aplicación, los modelos pueden ser catalizadores del proceso creativo al ayudar a superar bloqueos y explorar nuevas ideas con rapidez, o bien reemplazar por completo la labor del artista. Este dilema plantea importantes debates sobre la autoría, los derechos de propiedad intelectual y el futuro de diversas profesiones creativas. Actualmente, la relación entre la inteligencia artificial y los artistas se encuentra en un proceso de redefinición de su paradigma.

A pesar de sus impresionantes capacidades, los modelos generativos se enfrentan a retos muy significativos que se enumeran a continuación:

- Sensibilidad a los *prompts*: los modelos generativos son muy dependientes de los datos de entrada. Pequeñas variaciones en la información inicial pueden llevar a cambios drásticos en el resultado generado, lo que afecta la precisión de las respuestas y hace que sean menos fiables para usuarios sin experiencia previa.
- Alucinaciones e inconsistencias a largo plazo: persisten las dificultades para mantener la coherencia y consistencia en textos extensos debido a la capacidad finita de su memoria (ventana de contexto). También existe la posibilidad de que la salida se vuelva incongruente o que se genere información ficticia debido a factores internos o a la complejidad de la solicitud.
- Sesgos en los datos de entrenamiento: los modelos pueden perpetuar o amplificar prejuicios sociales presentes en los datos con los que fueron entrenados.

Para mitigar estos problemas utilizan las siguientes tecnicas:

- Generación aumentada por recuperación: esta técnica permite a los modelos consultar fuentes de datos externas y ampliar la información de la entrada con elementos que están fuera de su entrenamiento inicial o de su ventana de contexto inmediata. Esto reduce las alucinaciones y mejora la verosimilitud de la respuesta.
- Filtrado de datos de entrenamiento: se implementan procesos más rigurosos para limpiar y despolarizar los conjuntos de datos de entrenamiento, para minimizar la presencia de sesgos.
- *Fine-tuning* [4]: consiste en la adaptación de los modelos a tareas específicas a través de un proceso de reentrenamiento parcial.

[4] La idea de *fine-tuning* se popularizó enormemente con el desarrollo de los modelos de lenguaje pre-entrenados, siendo el paper de BERT[8] uno de los más influyentes al respecto.

- *Prompt engineering* [5]: se basa en la formulación precisa de instrucciones en la entrada para guiar la generación de la respuesta y obtener mejores resultados.

Este trabajo se fundamenta en este marco de conocimiento y se emplearon varias técnicas expuestas para mitigar el impacto de las limitaciones actuales de los modelos generativos.

1.4. Objetivos y alcance

El propósito del trabajo es desplegar un componente que aloje los modelos extensos de lenguaje con los que generar contenido narrativo a partir de una entrada. Este servidor también es capaz de administrar los modelos, añadir nuevos y elegir cuáles estarán activos.

A su vez, se dispone de un simple servidor web que actúa como la interfaz gráfica entre el usuario y la inteligencia artificial. Su función es recibir y mostrar la información al usuario, procesar sus instrucciones y el archivo adjunto y transformarlo en una instrucción personalizada según el servicio de generación narrativa solicitado. Esta instrucción es enviada al modelo de inteligencia artificial generativa, y su respuesta se reenvía al usuario.

Ambos servidores conforman un prototipo de pruebas que se entregará al cliente, junto con este documento, con el propósito de funcionar como estudio inicial y como base para futuros desarrollos dentro de su aplicación.

El alcance del trabajo incluye los siguientes elementos:

- Desarrollo de un servidor web.
- Descripción de los modelos extensos de lenguaje utilizados.
- Definición, implementación y descripción del *prompt engineering* para cada servicio.
- La integración de un conjunto esencial de servicios REST.

El alcance del trabajo no incluye:

- Diseño de la página que cumpla con los estándares más habituales.
- Securización del servidor web ni de ninguno de sus servicios REST.
- Rendimiento de los servicios que garanticen tiempos de respuesta cercanos a tiempo real.

[5] El concepto de *prompt engineering* se hizo fundamentalmente relevante y visible con la publicación del paper de GPT-3[12], que demostró las capacidades de aprendizaje *few-shot* e *in-context learning* mediante la formulación de instrucciones de texto.

Capítulo 2

Introducción específica

En este capítulo se profundiza en aquellos aspectos clave para el desarrollo de este trabajo. En primer lugar, se presentan los requerimientos de sistema y se amplía la información acerca de los modelos de inteligencia artificial que se han utilizado. Finalmente, se expone el conjunto de técnicas y herramientas que se han aplicado a dichos modelos.

2.1. Requerimientos del sistema

Para llevar a cabo este trabajo se identificaron una serie de requerimientos fundamentales. A continuación se listan aquellos que están directamente relacionados con la implementación:

1. Requerimientos del servidor:
    a) El servidor debe alojar y administrar la información relativa al *dataset* y la configuración del modelo LLM.
    b) El servidor debe contar con los *prompts* necesarios para especializar la respuesta de la Inteligencia Artificial.
    c) Al ser desplegado, el servidor deberá acceder al módulo LLM y utilizarlo en el procesamiento de peticiones entrantes.
    d) El servidor debe dar acceso a clientes externos a través del protocolo API REST.
    e) Los servicios REST deben aceptar entrada de texto en varios formatos: pdf, txt, word o texto plano en el cuerpo de la petición.
    f) Los servicios REST devolverán la respuesta en formato simple HTML para su cómoda visualización en un navegador.
    g) El prototipo del servidor debe de tener una disponibilidad del 100 % durante la demostración.
2. Requerimientos del módulo LLM:
    a) El módulo LLM debe aceptar entrada de texto y generar texto como salida.
    b) El módulo LLM, si el texto de entrada es legible, debe aportar una respuesta con un detalle y profundidad razonables, además de ser coherente con las instrucciones recibidas.

c) El módulo se ajustará a los prompts recibidos para que, con el mismo contexto, devuelva información enfocada en un aspecto específico de la narrativa.

d) El tiempo de respuesta del módulo LLM debe estar en un rango de tiempo razonable para un servicio REST (no más de 5 minutos).

Además, se identificaron requisitos adicionales relacionados con las pruebas, la documentación y los aspectos legales. Estos requisitos son vitales para asegurar la calidad del trabajo y garantizar el cumplimiento de los marcos legales pertinentes, incluyendo lo relativo a la propiedad intelectual, el uso de librerías de código abierto y la utilización de modelos con licencia libre o académica.

2.2. Modelos extensos de lenguaje

Para abordar los requisitos de generación de texto resultó fundamental la utilización de modelos extensos de lenguaje, comúnmente denominados *Large Language Models* (LLM). Estos modelos son entrenados sobre enormes volúmenes de datos textuales para especializarlos en predecir la secuencia de texto más probable dada una entrada previa. Esta habilidad les permite generar texto de forma coherente con el contexto al adaptarse al tono, estilo y contenido esperado.

Para conseguirlo, los LLM se basan en redes neuronales profundas que manejan miles de millones de parámetros (lo que justifica su clasificación como modelos "extensos") para identificar patrones complejos del lenguaje natural [6]. Estos modelos utilizan *tokens* como unidad básica de texto en su procesamiento. Los *tokens* pueden representar palabras completas, fragmentos de palabras o signos de puntuación, dependiendo del sistema de tokenización utilizado. Estos *tokens* se convierten posteriormente en vectores numéricos mediante una capa de *embedding*[13], que permite al modelo operar sobre ellos.

Los LLM modernos utilizan la arquitectura de *transformers*[6], un tipo de red neuronal que permite procesar secuencias de texto en paralelo y asignar diferentes niveles de relevancia (atención) a distintas partes del texto mediante un mecanismo llamado *self-attention*. Esta arquitectura supera notablemente a estructuras anteriores como *Long-Short Term Memory*[14] o *Gated Recurrent Unit*[15] en cuestiones de eficiencia y rendimiento.

Además de su capacidad de paralelización y memoria a largo plazo, los LLM basados en *transformers* operan mediante la predicción de la siguiente cadena de *tokens* a partir del contexto previo. En este proceso hay elementos que controlan la generacion, entre los que se destaca la temperatura[16]. La temperatura regula la aleatoriedad de los tokens: valores bajos tienden a generar respuestas deterministas, mientras que valores altos proporcionan respuestas más diversas y creativas[1]. Existen otros parámetros que controlan la variedad del texto, por ejemplo *top-k* y *top-p*[17].

A medida que los LLM aumentan en extensión de parámetros y profundidad de entrenamiento, empiezan a manifestar cualidades que no fueron explícitamente programadas. Entre estas capacidades destacan el razonamiento en múltiples pasos, traducción automática entre idiomas, capacidad de responder a preguntas

[1] Esto también aumenta las probabilidades de que esta respuesta sea incoherente, también llamada alucinación.

complejas y generación creativa y coherente de texto que va más allá de la simple predicción de cadenas de texto.

Por ejemplo, modelos como GPT-3[12] de OpenAI, PaLM[18] de Google y LLaMA[19] de Meta han demostrado un rendimiento notable en multitud de tareas complejas. Algunos ejemplos son realizar inferencias lógicas, resolver problemas matemáticos, resumir textos extensos y generar código a partir de descripciones en lenguaje natural. Esto no solo amplía el espectro de aplicaciones prácticas de los LLM, sino que también plantea nuevas líneas de investigación para comprender cómo el tamaño y la calidad del entrenamiento impactan en la adquisición de habilidades cognitivas sofisticadas.

2.3. Ingeniería de prompts

La ingeniería de *prompts* (*prompt engineering*) es una técnica fundamental en la optimización de la salida de los modelos extensos de lenguaje que se basa en guiar el proceso de razonamiento del modelo hacia un objetivo específico a través de un conjunto de entradas textuales. A diferencia de la programación tradicional, donde se expresa de forma explícita la lógica mediante el uso de código en un lenguaje de programación, el comportamiento de los LLM se guía mediante lenguaje natural.

Esta técnica ha emergido como una nueva disciplina en la que intervienen conceptos linguísticos y computacionales. La forma en la que se redacta una instrucción puede influir de forma notable en la coherencia, relevancia, creatividad y precisión de las respuestas.

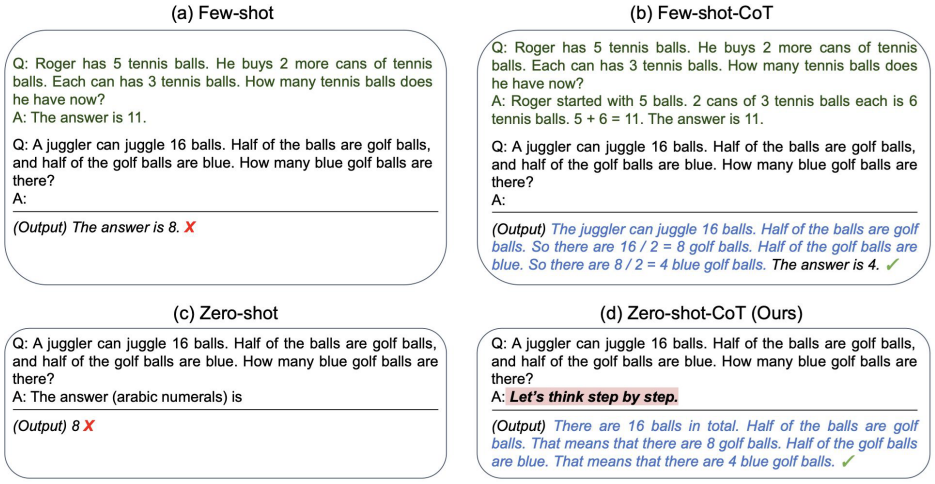

FIGURA 2.1. Ejemplo de uso de técnicas de prompting.

Entre las estrategias comunes de ingeniería de prompts se encuentran el *zero/few-shot prompting*[20, 21], que se apoya en ejemplos en el prompts para guiar la generación; el *chain of thought prompting*[22] invita al modelo a razonar explícitamente los pasos intermedios antes de llegar a la conclusión; o *prompt chaining*[23] que invita al modelo a realizar un análisis previo sobre el contexto o instrucciones para

enfocar la salida. En la figura 2.1[2] se ilustra cómo estas estrategias de instrucción afectan directamente la salida generada por el modelo. Cabe destacar que las distintas técnicas de prompting descritas no son excluyentes entre sí, sino que pueden combinarse de manera complementaria, potenciando así la capacidad de razonamiento y la precisión de los LLM en tareas complejas.

A su vez, la ingeniería de *prompts* ha demostrado ser fundamental en aquellos contextos en los que el *fine-tuning* no es viable, ya que hay muchos modelos cerrados que solo son accesibles a través de una API y la capacidad de modificar su comportamiento sin reentrenamiento se vuelve crucial. Por lo tanto, esta técnica se convierte en una herramienta práctica, eficiente y cada vez más sofisticada para adaptar modelos generalistas a tareas específicas.

2.4. Ajuste fino

El ajuste fino (*fine-tuning*)[25] es una técnica de entrenamiento de modelos extensos de lenguaje que permite adaptar un modelo previamente entrenado a una tarea en específico. A diferencia del entrenamiento desde cero, esta técnica parte de una base y lo especializa utilizando un conjunto mucho más pequeño y específico de datos. De este modo se reduce significativamente el coste computacional a la vez que se aprovecha la capacidad predictiva del modelo del que se parte.

Para llevar a cabo el ajuste fino, se conserva la estructura y los parámetros previamente entrenados del modelo base, evitando así la necesidad de un entrenamiento completo desde cero. En la práctica, el proceso de fine-tuning suele centrarse en modificar únicamente una parte reducida del modelo, como las capas superiores, que son las más cercanas a la salida y, por tanto, más fácilmente adaptables a tareas específicas. Alternativamente, pueden integrarse nuevas capas que se entrenan sin alterar el resto del modelo. Esta aproximación permite preservar los conocimientos generales adquiridos durante el preentrenamiento, mientras se optimiza la capacidad del modelo para tareas concretas.

Además, en escenarios donde los recursos computacionales son limitados, se utilizan técnicas de ajuste fino eficiente (parameter-efficient fine-tuning, PEFT), como LoRA (*Low-Rank Adaptation*)[26], *prefix tuning*[27] o *prompt tuning*. Estas estrategias reducen la cantidad de parámetros que deben entrenarse, lo que no solo disminuye el tiempo de ajuste y el consumo de memoria, sino que también facilita la reutilización del modelo base en múltiples contextos sin interferencia entre tareas.

Al adaptar un modelo generalista a contextos específicos, se logra una mejora sustancial en la precisión, relevancia y sensibilidad del modelo frente a matices del lenguaje propios del área de aplicación. No obstante, es fundamental contar con datos de alta calidad y bien etiquetados para evitar la sobreespecialización o la introducción de sesgos.

[2] Imagen obtenida del material del curso de especializacion en inteligencia artificial de la FIUBA, materia de *LLM* [24]

2.5. LM Studio y otras tecnologías

LM Studio[28] es una plataforma de código abierto diseñada para facilitar la interacción y despliegue de modelos extensos de lenguaje (LLM) de manera local, permitiendo a los usuarios ejecutar y experimentar con modelos sin depender de servicios en la nube. Esta herramienta destaca por su integración sencilla, soporte para múltiples formatos de modelos y una interfaz amigable.

Una de las ventajas clave de LM Studio es su capacidad para manejar modelos grandes y complejos con eficiencia, ofreciendo funcionalidades como la gestión de memoria optimizada, soporte para cuantización y carga progresiva de modelos. Esto permite que usuarios con recursos limitados puedan aprovechar la potencia de los LLM sin necesidad de infraestructura costosa. Además, LM Studio facilita la personalización de modelos mediante interfaces accesibles, lo que la convierte en una opción popular tanto para investigadores como para desarrolladores que desean incorporar inteligencia artificial avanzada en sus proyectos.

Para la implementación del *backend* de la aplicación que interactúa con LM Studio, se utilizó *Python*, un lenguaje de programación ampliamente adoptado en el ámbito de la inteligencia artificial por su simplicidad y la gran cantidad de bibliotecas disponibles. Se empleó la librería de *FastAPI*[29] en al construcción del servidor web gracias a su sintaxis intuitiva, lo que facilitó la creación de *endpoints* eficientes para la comunicación entre el usuario y el modelo.

Por otro lado, *PyTorch*[30] es la biblioteca de referencia para el desarrollo y entrenamiento de modelos de aprendizaje profundo, incluyendo los LLM. Proporciona una interfaz dinámica y flexible que permite tanto el entrenamiento como la inferencia eficiente en hardware acelerado, y es compatible con múltiples plataformas.

Capítulo 3

Diseño e implementación

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

3.1. Análisis del software

La idea de esta sección es resaltar los problemas encontrados, los criterios utilizados y la justificación de las decisiones que se hayan tomado.

Se puede agregar código o pseudocódigo dentro de un entorno lstlisting con el siguiente código:

```
\begin{lstlisting}[caption= "un epígrafe descriptivo"]
 las líneas de código irían aquí...
\end{lstlisting}
```

A modo de ejemplo:

```
#define MAX_SENSOR_NUMBER 3
#define MAX_ALARM_NUMBER  6
#define MAX_ACTUATOR_NUMBER 6

uint32_t sensorValue[MAX_SENSOR_NUMBER];
FunctionalState alarmControl[MAX_ALARM_NUMBER]; //ENABLE or DISABLE
state_t alarmState[MAX_ALARM_NUMBER];           //ON or OFF
state_t actuatorState[MAX_ACTUATOR_NUMBER];     //ON or OFF

void vControl() {

  initGlobalVariables();

  period = 500 ms;

  while(1) {

    ticks = xTaskGetTickCount();

    updateSensors();

    updateAlarms();

    controlActuators();

    vTaskDelayUntil(&ticks, period);
  }
```

```
28 }
```

CÓDIGO 3.1. Pseudocódigo del lazo principal de control.

Capítulo 4

Ensayos y resultados

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

4.1. Pruebas funcionales del hardware

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

Capítulo 5

Conclusiones

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

5.1. Conclusiones generales

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

En esta sección no se deben incluir ni tablas ni gráficos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Próximos pasos

Acá se indica cómo se podría continuar el trabajo más adelante.

Bibliografía

[1] Tomás Muriel. *Narrativa*. 2 de oct. de 2023. URL: https://www.significados.com/narrativa/ (visitado 24-05-2025).

[2] Jeff VanderMeer. *Wonderbook: The Illustrated Guide to Creating Imaginative Fiction*. Abrams Image, 2013. ISBN: 978-1-4197-0681-7.

[3] J.R.R. Tolkien. *The Letters of J.R.R. Tolkien*. Houghton Mifflin, 1981.

[4] *Critical Match: Plataforma para juegos de mesa*. https://www.criticalmatch.app/. Plataforma enfocada en juegos de mesa como rol, cartas coleccionables y wargames. (Visitado 24-05-2025).

[5] Critical Match UG. *Critical Match: Encuentra mesas de rol*. Google Play Store. Versión de la aplicación móvil. 2025. URL: https://play.google.com/store/apps/details?id=com.criticalmatch (visitado 24-05-2025).

[6] Ashish Vaswani et al. «Attention is all you need». En: (). arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[7] Alec Radford et al. «Improving Language Understanding by Generative Pre-Training». En: *OpenAI* (jun. de 2018). Technical Report. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf (visitado 24-05-2025).

[8] Jacob Devlin et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». En: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (2019), págs. 4171-4186. DOI: 10.18653/v1/N19-1152. URL: https://arxiv.org/abs/1810.04805.

[9] OpenAI. *ChatGPT: Optimizing Language Models for Dialogue*. Blog post. Nov. de 2022. URL: https://openai.com/blog/chatgpt/ (visitado 24-05-2025).

[10] Gemini Team. «Gemini: A Family of Highly Capable Multimodal Models». En: *arXiv preprint arXiv:2312.11805* (2023). URL: https://arxiv.org/abs/2312.11805.

[11] Meta AI. *About Meta AI*. Website. Página oficial de Meta AI. URL: https://ai.meta.com/ (visitado 24-05-2025).

[12] Tom B. Brown et al. «Language Models are Few-Shot Learners». En: *Advances in Neural Information Processing Systems* 33 (2020), págs. 1877-1901. URL: https://proceedings.neurips.cc/paper/2020/file/145749ea3f59e0b57574be084196c703-Paper.pdf.

[13] Tomas Mikolov et al. «Efficient Estimation of Word Representations in Vector Space». En: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2013.

[14] Sepp Hochreiter y Jürgen Schmidhuber. «Long short-term memory». En: *Neural computation* 9.8 (1997), págs. 1735-1780.

[15] Kyunghyun Cho et al. «Learning phrase representations using RNN encoder-decoder for statistical machine translation». En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, págs. 1724-1734.

[16] Alec Radford et al. «Language models are unsupervised multitask learners». En: *OpenAI blog* 1.8 (2019), pág. 9.
[17] Ari Holtzman et al. «The curious case of neural text degeneration». En: *arXiv preprint arXiv:1904.09751* (2019).
[18] Aakanksha Chowdhery et al. «PaLM: Scaling Language Modeling with Pathways». En: *arXiv preprint arXiv:2204.02311* (2022).
[19] Hugo Touvron et al. «LLaMA: Open and Efficient Foundation Language Models». En: *arXiv preprint arXiv:2302.13971* (2023).
[20] Tom B Brown et al. «Language models are few-shot learners». En: *Advances in neural information processing systems* 33 (2020), págs. 1877-1901.
[21] Takeshi Kojima et al. «Large language models are zero-shot reasoners». En: *arXiv preprint arXiv:2205.11916* (2022).
[22] Jason Wei et al. «Chain of thought prompting elicits reasoning in large language models». En: *arXiv preprint arXiv:2201.11903* (2022).
[23] Prompting Guide. *Prompt Chaining*. https://www.promptingguide.ai/techniques/prompt_chaining. 2023.
[24] Esp. Ing Abraham Rodriguez FIUBA y Esp. Ing Ezequiel Guinsburg FIUBA. *MoEs, Prompting y evaluación*. https://github.com/FIUBA-Posgrado-Inteligencia-Artificial/CEIA-LLMIAG/blob/main/ClaseIV/Clase_IV_MoEs.pdf. Material de clase, disponible en GitHub. 2024.
[25] Jeremy Howard y Sebastian Ruder. «Universal language model fine-tuning for text classification». En: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018), págs. 328-339.
[26] Edward J Hu et al. «LoRA: Low-Rank Adaptation of Large Language Models». En: *arXiv preprint arXiv:2106.09685* (2021).
[27] Xiang Lisa Li y Percy Liang. «Prefix-Tuning: Optimizing Continuous Prompts for Generation». En: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. 2021, págs. 4582-4597.
[28] Nomic AI. *LM Studio*. https://lmstudio.ai/. Accessed: 2025-05-26.
[29] Sebastián Ramírez. *FastAPI*. https://fastapi.tiangolo.com/. Accessed: 2025-05-26.
[30] PyTorch Contributors. *PyTorch*. https://pytorch.org/. Accessed: 2025-05-26.