Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Inteligencia artificial

Estudio comparativo de técnicas PEFT sobre modelos LLM para la generación de SQL a partir de lenguaje natural en entornos con recursos limitados

Trabajo fin de estudio presentado por:	Carrasco Martínez, Antoni Molera Gómez, Ivan
Tipo de trabajo:	Comparativa de soluciones
Director/a:	Xiomara Patricia Blanco Valencia
Fecha:	10/07/2025

Resumen

Este trabajo presenta una evaluación comparativa de técnicas de ajuste fino eficiente en parámetros (PEFT) aplicadas a modelos de lenguaje de gran tamaño (LLM), con énfasis en su rendimiento para la tarea específica de generación de consultas SQL a partir de lenguaje natural (NL-to-SQL) en entornos con recursos computacionales limitados. Se analizan cuatro técnicas PEFT destacadas: LoRA, QLoRA, Prefix Tuning e $(IA)^3$, utilizando como modelos base GPT-2 y LLAMA-7B. La evaluación considera tanto aspectos relacionados con la eficiencia computacional (tiempo de entrenamiento, consumo de memoria GPU y utilización de recursos) como métricas de rendimiento en la tarea de traducción NL-to-SQL (función de pérdida en entrenamiento y evaluación, convergencia, generalización y tiempos de inferencia). Este análisis concluye que técnicas como QLoRA y LoRA constituyen soluciones preferentes para entornos restringidos computacionalmente, proporcionando resultados robustos y eficientes, destacando por su bajo consumo de memoria GPU y rápida convergencia en la tarea de generación de consultas SQL desde lenguaje natural. Por otra parte, Prefix Tuning e $(IA)^3$ presentan tiempos de entrenamiento significativamente mayores y requieren ajustes adicionales para alcanzar rendimientos comparables, haciendo su adopción menos viable en el tipo de escenario descrito sin ajustes profundos en hiperparámetros.

Palabras clave: PEFT, LLM, NL-to-SQL, fine-tuning eficiente, recursos computacionales limitados.

Abstract

This study presents a comparative evaluation of parameter-efficient fine-tuning techniques (PEFT) applied to large language models (LLMs), with an emphasis on their performance for the specific task of generating SQL queries from natural language (NL-to-SQL) in computationally constrained environments. Four prominent PEFT techniques are analyzed: LoRA, QLoRA, Prefix Tuning, and $(IA)^3$, using GPT-2 and LLAMA-7B as base models. The evaluation considers both aspects related to computational efficiency (training time, GPU memory consumption, and resource utilization) and performance metrics in the NL-to-SQL translation task (loss function in training and evaluation, convergence, generalization, and inference times). This analysis concludes that techniques such as QLoRA and LoRA constitute preferable solutions for computationally constrained environments, providing robust and efficient results, standing out for their low GPU memory consumption and fast convergence in the task of generating SQL queries from natural language. On the other hand, Prefix Tuning and $(AI)^3$ present significantly longer training times and require additional tuning to achieve comparable performance, making their adoption less viable in the described scenario without a deeper hyperparameter tuning.

Keywords: PEFT, LLM, NL-to-SQL, efficient fine-tuning, limited computational resources.

Índice de contenidos

Índice de figuras

Índice de tablas

Organización del trabajo en grupo

La elaboración del trabajo ha sido mayormente una tarea coral, con un proceso constante de interacción y revisión mutua de cada avance, de forma que ambos integrantes han contribuido activamente en todas las secciones del documento. No obstante, en la fase de implementación de la comparativa se estableció una división puntual de responsabilidades: Antoni Carrasco se focalizó en la puesta en marcha del entorno de pruebas y su configuración técnica, mientras que Ivan Molera se centró en el análisis detallado del conjunto de datos empleado (WikiSQL) y su adecuación al objetivo del proyecto. Esta división permitió optimizar tiempos y profundizar en cada área con un enfoque más específico, manteniendo siempre una coordinación estrecha mediante reuniones periódicas y revisiones conjuntas.

Partes que aborda el TFE

Cada integrante del grupo ha asumido tareas que, individualmente, han contribuido a la consecución del TFE. Entre estas destacan la selección crítica y evaluación de técnicas PEFT sobre modelos LLM, el diseño y configuración del entorno de pruebas, el análisis detallado del dataset de evaluación y la sistematización comparativa de resultados. La combinación de estos trabajos en un único proyecto ha proporcionado una visión global que integra tanto el enfoque experimental como el análisis crítico, fortaleciendo el carácter multidisciplinar del estudio.

Distribución y estructura de la memoria

Tabla 1. *Organización del trabajo en grupo*

Organización del trabajo en grupo - Desarrollo de la memoria	
Apartado de la memoria	**Responsables**
Introducción	Carrasco, Molera
Contexto y estado del arte	Carrasco, Molera
Objetivos y metodología de trabajo	Carrasco, Molera
Planteamiento de la comparativa	Carrasco, Molera
Definición del problema y objetivos de negocio	Carrasco, Molera
Comprensión del dominio y los datos	Carrasco, Molera
Preparación de los datos	Carrasco, Molera
Modelado	Carrasco, Molera
Evaluación	Carrasco, Molera
Conclusiones y trabajo futuro	Carrasco, Molera

Fuente: Elaboración propia.

Objetivo del TFE desde el punto de vista de la adquisición de conocimientos

Desde el punto de vista de la adquisición de conocimientos, este trabajo de fin de estudios incorpora diversos segmentos claramente diferenciados que, en sí mismos, podrían constituir propuestas de investigación individuales. A través de esta aproximación multidisciplinar, los autores abordan aspectos tanto teóricos como empíricos relacionados con las técnicas de ajuste fino eficiente en parámetros (PEFT) y sobre modelos de lenguaje de gran tamaño (LLM), específicamente enfocados en la tarea de generación de consultas SQL a partir de lenguaje natural en contextos con recursos computacionales limitados.

Este enfoque ha permitido adquirir competencias transversales y específicas, incluyendo el análisis comparativo de técnicas avanzadas como LoRA, QLoRA, Prefix Tuning e $(IA)^3$, además

del uso y evaluación de modelos base como GPT-2 y LLAMA-7B. Además, el carácter grupal del trabajo ha conferido al proyecto un enfoque holístico que va más allá del ámbito académico, permitiendo un resultado aplicable en contextos profesionales reales, especialmente en entornos que requieren soluciones eficientes y escalables en términos de recursos computacionales. De este modo, no solo se ha profundizado en aspectos teóricos y técnicos del ajuste fino de modelos, sino que también se han desarrollado habilidades prácticas aplicables directamente en un futuro entorno laboral.

Mecanismos de coordinación empleados

La coordinación efectiva ha sido un componente crucial en el desarrollo exitoso de este trabajo, especialmente considerando la complejidad inherente a un proyecto que involucra múltiples áreas técnicas y de investigación. Los mecanismos empleados han consistido principalmente en reuniones periódicas, tanto en formato online (utilizando Google Meet) como en sesiones presenciales. Estas reuniones han servido para sincronizar el progreso individual de las tareas asignadas, revisar y discutir colectivamente los avances alcanzados, identificar obstáculos y debatir soluciones conjuntas para enfocar adecuadamente el desarrollo del trabajo.

Cabe mencionar que no se han aplicado frameworks de trabajo más avanzados como Scrum debido al reducido tamaño del equipo de trabajo, compuesto únicamente por dos personas, lo que ha permitido una comunicación directa más eficiente y efectiva sin necesidad de herramientas adicionales de gestión de proyectos.

Adicionalmente, se ha mantenido una comunicación continua con la tutora del proyecto, Xiomara Patricia Blanco Valencia, para recibir retroalimentación crítica y constructiva, permitiendo así ajustar y mejorar continuamente tanto los documentos intermedios como el enfoque general del trabajo. Estas interacciones no solo han facilitado la coordinación del trabajo, sino que han permitido llevar a cabo, de manera efectiva, la labor de colaboración constante y de retroalimentación iterativa, esenciales para la obtención de resultados sólidos y coherentes.

1. Introducción

1.1. Motivación

La traducción automática del lenguaje natural a consultas SQL (NL-to-SQL) es una tarea esencial para posibilitar una interacción fluida de usuarios no expertos (o con pocos conocimientos técnicos) con bases de datos relacionales. La dificultad inherente a este tipo de tarea radica en la complejidad del lenguaje natural, en la complejidad del modelo de datos aplicable al dominio de conocimiento, y en la necesidad de generar consultas SQL precisas y optimizadas. Para este tipo de tareas se requiere la aplicación de modelos de lenguaje de gran tamaño (LLM), cuyo entrenamiento y ajuste fino implican la necesidad de disponer de grandes recursos computacionales, restringiendo su uso a organizaciones con acceso a infraestructuras avanzadas.

La motivación principal de este trabajo es abordar precisamente esta limitación, evaluando técnicas avanzadas de ajuste fino eficiente en parámetros (PEFT), las cuales permitan adaptar modelos LLM pre-entrenados a tareas específicas, utilizando recursos computacionales limitados. Para este estudio comparativo se han analizado técnicas como LoRA, QLoRA, Prefix Tuning, $(IA)^3$ y BitFit, que pueden reducir significativamente la carga computacional requerida para un ajuste fino de modelos manteniendo un rendimiento competitivo. La relevancia de esta investigación se centra en aportar conocimiento práctico y sistematizado sobre cómo emplear eficazmente modelos complejos en entornos limitados en recursos, promoviendo la democratización del acceso a tecnologías avanzadas de procesamiento del lenguaje natural.

1.2. Planteamiento del trabajo

El problema detectado reside en las restricciones de infraestructura computacional existentes para la aplicación efectiva de modelos de lenguaje de gran tamaño en la generación automática de consultas SQL a partir del lenguaje natural, en base a un modelo de datos relacional. Esta limitación reduce notablemente la capacidad de las organizaciones con recursos limitados para adoptar tecnologías avanzadas de inteligencia artificial.

Para encontrar una solución a este problema, se propone aplicar un subconjunto de técnicas PEFT sobre dos modelos base (GPT-2 y LLaMA-7B), y evaluar de forma comparativa los resultados obtenidos con las técnicas seleccionadas. Esta selección se basa en la relevancia de las técnicas y en su distinta aproximación a la tarea de ajuste fino de modelos, para así proporcionar una visión amplia a la comparativa. El objetivo general es determinar cuál de estas técnicas ofrece el mejor equilibrio entre eficiencia computacional y rendimiento en tareas de traducción NL-to-SQL. Los objetivos específicos incluyen medir y analizar el consumo de recursos computacionales (tiempo de entrenamiento, memoria GPU, tiempo de inferencia, etc), así como evaluar la precisión y capacidad de generalización de las consultas SQL generadas por los modelos ajustados mediante cada una de las técnicas. Este análisis permitirá identificar claramente las ventajas y limitaciones de cada técnica, ofreciendo una base sólida para su adopción en diversos contextos.

1.3. Estructura del trabajo

La estructura de este trabajo está organizada en cinco capítulos principales:

- **Capítulo 1: Introducción.** Se presenta la motivación del estudio, la problemática que se busca resolver y el planteamiento general del trabajo, además de describir brevemente la estructura del documento.

- **Capítulo 2: Contexto y estado del arte. E**n este capítulo se revisa críticamente la evolución histórica, la clasificación actualizada y las características destacadas de las técnicas PEFT. Además, se analiza su aplicabilidad en entornos con recursos computacionales limitados, proporcionando una visión clara del marco teórico en el que se sustenta esta investigación.

- **Capítulo 3: Objetivos y metodología de trabajo.** Se definen de manera detallada los objetivos generales y específicos del estudio, junto con la metodología empleada para el desarrollo de la comparativa entre las técnicas seleccionadas. En este capítulo también se describen las herramientas y recursos utilizados, así como los criterios y procedimientos de evaluación implementados.

- **Capítulo 4: Planteamiento de la comparativa.** En este capítulo se describe en profundidad el contexto experimental, detallando la selección razonada de los modelos de lenguaje (GPT-2 y LLAMA-7B), así como de las técnicas de ajuste fino elegidas. Se analiza también el conjunto de datos utilizado para los experimentos (WikiSQL), las métricas empleadas para la evaluación y el análisis de los resultados obtenidos.

- **Capítulo 5: Conclusiones y trabajo futuro.** Se resumen los principales hallazgos del estudio, destacando claramente qué técnicas proporcionan el mejor equilibrio entre eficiencia y rendimiento. Asimismo, se plantean algunas recomendaciones para investigaciones futuras relacionadas con la optimización y aplicación efectiva de técnicas PEFT en contextos con recursos computacionales limitados.

2. Contexto y estado del arte

2.1. Contexto

En los últimos años, los modelos de lenguaje de gran tamaño (***Large Language Models*** o ***LLM***) han transformado significativamente el campo del procesamiento del lenguaje natural (***Natural Language Processing***, ***NLP***), impulsando avances notables en tareas como la generación de texto, el resumen automático, la traducción y el análisis semántico. Estos modelos, entrenados sobre grandes cantidades de datos y compuestos por miles de millones de parámetros, han demostrado una capacidad notable para generalizar y adaptarse a múltiples tareas con mínima intervención humana.

El desarrollo de los LLM se encuentra estrechamente vinculado con la introducción de la arquitectura Transformer, presentada por Vaswani et al. en su artículo "Attention is All You Need" (Vaswani et al., 2017). Esta arquitectura revolucionó el paradigma del aprendizaje profundo en NLP al reemplazar los mecanismos secuenciales de las redes recurrentes por un sistema de atención que permite procesar los datos en paralelo, mejorando significativamente tanto el rendimiento como la escalabilidad del entrenamiento de modelos. Desde entonces, los Transformers han servido como base para el desarrollo de modelos como BERT, GPT, T5 y muchos otros, conformando el núcleo de los actuales LLM.

No obstante, el ajuste fino (***fine-tuning***) de estos modelos presenta importantes desafíos, especialmente por los elevados recursos computacionales que requiere. Ajustar todos los parámetros de un modelo de gran escala implica una demanda significativa de memoria, potencia de cómputo y tiempo de entrenamiento, lo cual puede ser prohibitivo en entornos con infraestructura limitada.

Este problema es particularmente crítico para organizaciones sin acceso a infraestructuras avanzadas o que prefieren evitar la dependencia de servicios en la nube por motivos relacionados con privacidad, normativas internas o estrategias corporativas. En consecuencia, existe un interés creciente en técnicas de ajuste fino eficientes en parámetros (***parameter-efficient fine-tuning***, ***PEFT***), las cuales permiten adaptar modelos LLM a contextos específicos manteniendo congelada la mayoría de sus capas y entrenando únicamente un subconjunto

reducido de parámetros. Estas técnicas, además de optimizar el uso de recursos computacionales, también se alinean con los principios de la inteligencia artificial sostenible, una tendencia cada vez más relevante en la actualidad, y facilitan así la implementación de modelos adaptados en entornos con recursos computacionales limitados (***on-premise***) (Yang et al., 2024).

En paralelo al ajuste fino, también han surgido estrategias orientadas a enriquecer la salida de los LLM mediante el uso de fuentes externas de conocimiento. Un enfoque destacado en este ámbito es el ***Retrieval-Augmented Generation*** (***RAG***) (P. Lewis et al., 2020), una técnica que combina modelos generativos con módulos de recuperación de información. En lugar de depender únicamente del conocimiento codificado durante el entrenamiento, un sistema RAG consulta una base de datos o corpus documental relevante durante la inferencia, integrando la información recuperada en la generación de la respuesta final. Esta arquitectura híbrida permite mejorar la precisión, actualidad y trazabilidad de las respuestas generadas, siendo especialmente útil en escenarios donde la actualización frecuente del modelo completo no es viable. No obstante, dado que estas técnicas se centran en la etapa de inferencia y no en el ajuste de los parámetros del modelo base, su análisis detallado queda fuera del alcance del presente trabajo.

Los objetivos generales del presente capítulo incluyen:

- Identificar y describir las principales técnicas de ***fine-tuning*** eficientes en parámetros aplicadas a modelos ***LLM***.
- Analizar la evolución histórica y los problemas que cada técnica pretende resolver.
- Evaluar la aplicabilidad práctica en escenarios reales con recursos computacionales restringidos.

2.2. Evolución y clasificación de las técnicas de ajuste fino PEFT

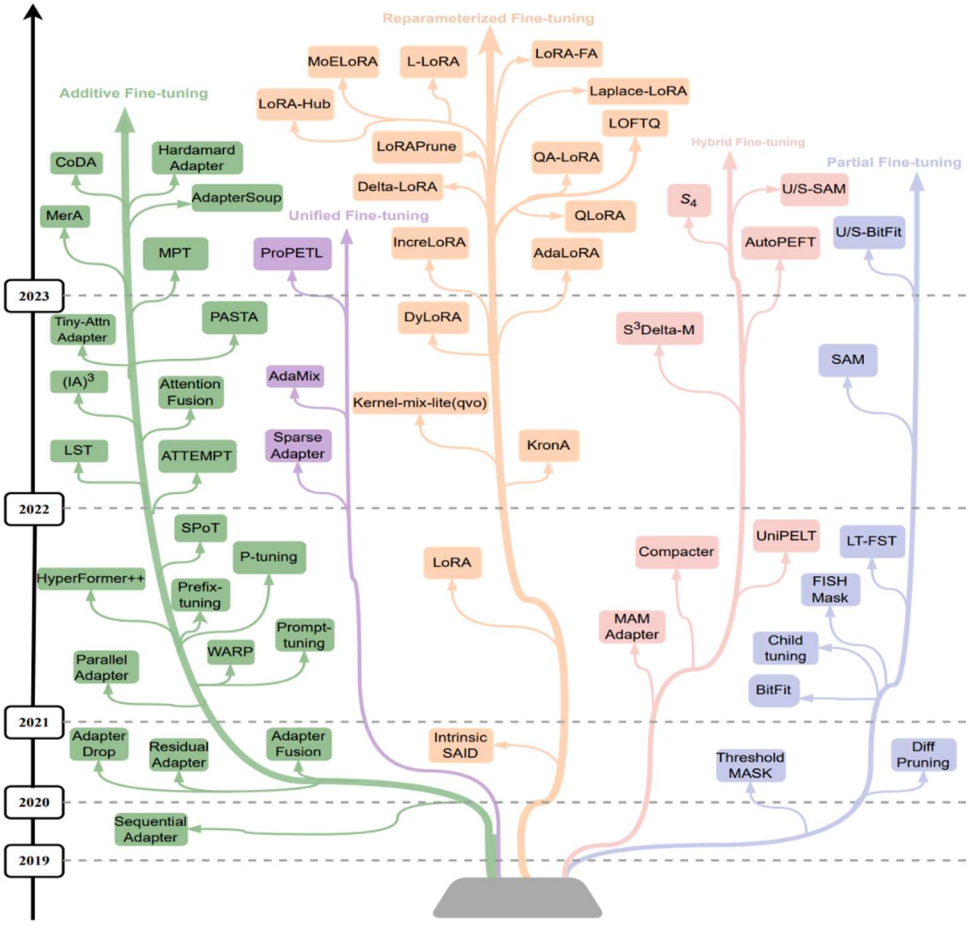
El ajuste fino eficiente en parámetros (PEFT) ha emergido como una solución crucial frente a los elevados costes computacionales asociados al ajuste completo de modelos de lenguaje de gran tamaño. A medida que los LLM aumentan su número de parámetros hasta alcanzar cientos de miles de millones, resulta inviable ajustar todos sus pesos en escenarios con recursos computacionales limitados. En este contexto, los métodos PEFT permiten adaptar

modelos preentrenados sin modificar la mayoría de sus parámetros, minimizando así el consumo de memoria y recursos de cómputo (Xu et al., 2023).

2.3. Evolución de las técnicas PEFT

La evolución de los métodos PEFT ha sido rápida y continua desde 2019, como se muestra en la Figura 1 (Los modelos agrupados en una misma rama comparten características comunes y la posición vertical indica el año de publicación). Inicialmente centrados en enfoques aditivos como los Adapters, con el tiempo se diversificaron hacia métodos basados en selección, reparametrización y combinaciones híbridas (Lialin et al., 2023). Esta evolución refleja tanto la necesidad de reducir los costes computacionales como de mejorar la modularidad y la capacidad de reutilización entre tareas.

Figura 1. *Evolución temporal de los métodos PEFT*



Fuente: (Xu et al., 2023).

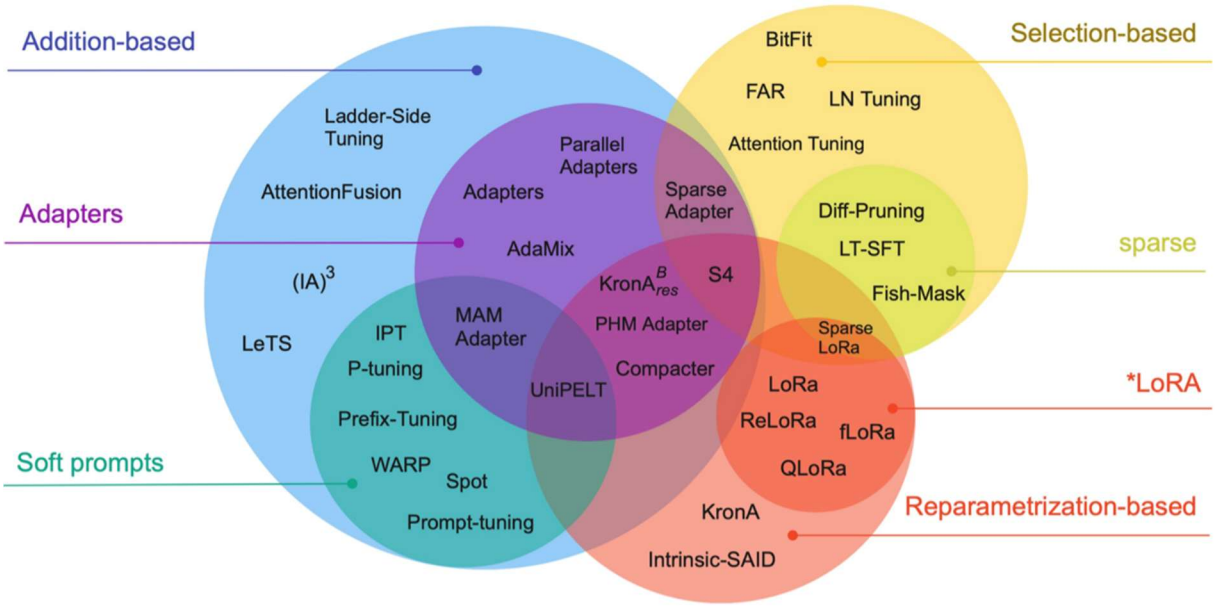
2.4. Clasificación de las técnicas PEFT

Se distinguen tres categorías principales en la clasificación de métodos PEFT: métodos basados en adición, en selección y en reparametrización (Ben Zaken et al., 2023).

- ***Métodos basados en adición (Addition-based)***: incorporan nuevos parámetros entrenables al modelo, como los módulos adapter o los soft prompts. Dentro de esta categoría destacan los enfoques tipo adaptador, que añaden capas entre los bloques del Transformer, y los soft prompts, que introducen vectores entrenables en las entradas o representaciones internas del modelo.
- ***Métodos basados en selección (Selection-based)*:** ajustan solo un subconjunto de los parámetros existentes, como los bias (en BitFit) o máscaras aplicadas a los pesos (como en Diff Pruning).
- ***Métodos basados en reparametrización (Reparametrization-based)***: reestructuran los parámetros mediante descomposiciones de bajo rango o productos de Kronecker, reduciendo así el número de parámetros necesarios para la adaptación. LoRA y sus variantes son ejemplos destacados.

Esta clasificación se representa gráficamente en la siguiente figura, donde se observan las intersecciones entre categorías, así como la diversidad de enfoques actuales.

Figura 2. *Taxonomía de los métodos de ajuste fino eficientes en parámetros (PEFT)*



Fuente: (Ben Zaken et al., 2023).

La clasificación y evolución de estos métodos proporciona un marco conceptual sólido para evaluar su aplicabilidad en contextos prácticos. La distinción entre tipos de técnicas permite seleccionar la estrategia más adecuada en función de los recursos disponibles, el tipo de tarea y el tamaño del modelo base.

2.5. Técnicas de ajuste fino PEFT

La selección de técnicas PEFT descritas en esta sección se fundamenta en tres criterios principales derivados de estudios recientes (Ben Zaken et al., 2023) y (Xu et al., 2023). Primero, se priorizaron aquellas técnicas que requieren ajustar un porcentaje muy limitado de parámetros, generalmente inferior al 5%, debido a que estos métodos ofrecen una notable reducción en el uso de memoria y recursos computacionales, esenciales en entornos con limitaciones técnicas y presupuestarias. Técnicas como LoRA, BitFit, $(IA)^3$ y QLoRA destacan particularmente por esta característica.

En segundo lugar, se escogieron técnicas que no modifican los pesos del modelo base original, garantizando así la integridad del modelo preentrenado y facilitando la reutilización del mismo modelo para diferentes tareas sin necesidad de almacenamiento adicional significativo. La modularidad y capacidad de adaptación en múltiples tareas representadas por AdapterFusion, Prefix Tuning y P-Tuning v2 también fueron cruciales para su selección. Estas características favorecen la aplicación práctica de los métodos PEFT en contextos donde los recursos son limitados.

Finalmente, se seleccionaron las técnicas considerando el criterio de la sobrecarga en inferencia (inference overhead), es decir, el coste adicional en tiempo, memoria o recursos computacionales que se produce durante la fase de inferencia debido a modificaciones introducidas en el modelo durante su ajuste fino. Este aspecto es particularmente relevante en entornos con recursos limitados, dado que un elevado inference overhead podría ralentizar considerablemente la respuesta del modelo, aumentar significativamente los requisitos de GPU o CPU, o dificultar su escalabilidad. En consecuencia, se priorizaron aquellas técnicas PEFT que presentan una sobrecarga reducida en inferencia, facilitando su implementación efectiva en condiciones operativas restringidas.

Tabla 2. *Resumen ampliado de técnicas PEFT (precisión relativa, eficiencia y complejidad)*

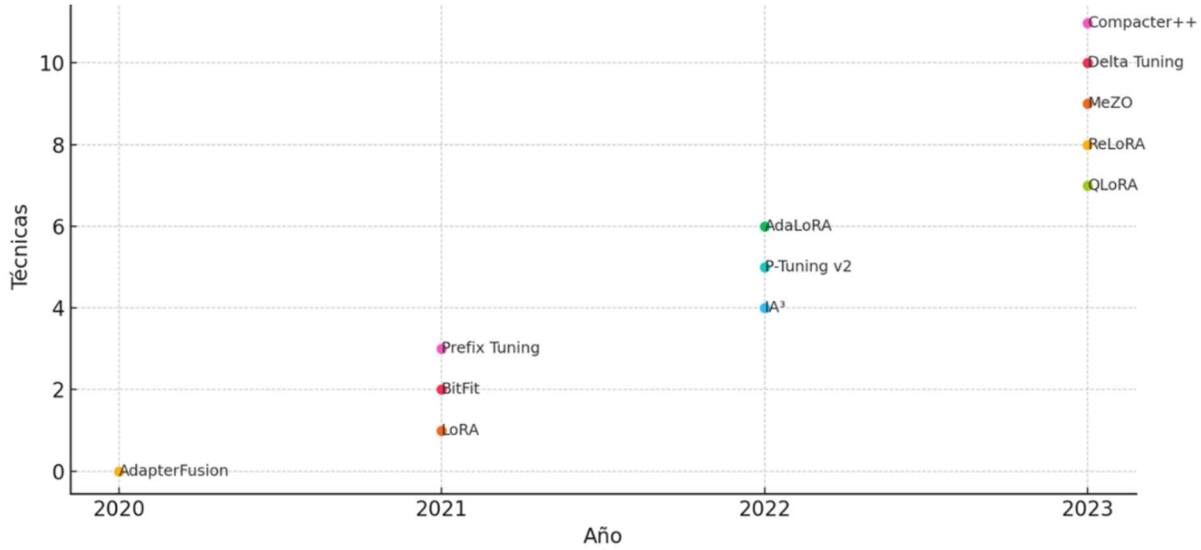
Técnica	Año	Tipo de optimización	Parámetros entrenables (%)	Modifica pesos base?	Precisión	Eficiencia	Complejidad	Inference overhead
AdapterFusion	2020	Fusión de adaptadores	~1–5%	No	★★★★	★★★	◍ Moderada	Sí
LoRA	2021	Adaptación de bajo rango	~1%	No	★★★★	★★★★★	◍ Media	No
BitFit	2021	Ajuste de bias	~0.1%	No	★★	★★★★★	◍ Baja	No
Prefix Tuning	2021	Prompts continuos	~0.1%	No	★★★★	★★★★	◍ Media	Sí
(IA)³	2022	Reescalado de activaciones	~0.01%	No	★★★★	★★★★	◍ Moderada	Sí
P-Tuning v2	2022	Prompts en múltiples capas	~0.1–3%	No	★★★★	★★★★	◍ Media	Sí
AdaLoRA	2022	Adaptación dinámica de LoRA	~1%	No	★★★★	★★★★	◍ Moderada	No
QLoRA	2023	Cuantización + LoRA	~1%	No	★★★★	★★★★★	◍ Moderada	No
ReLoRA	2023	Reinicialización periódica	~1%	No	★★★★	★★★★	◍ Media	No

MeZO	2023	Zero-order	~1%	No	★★	★★★★	Moderada	No
Delta Tuning	2023	Deltas parametrizadas	~1%	No	★★★★	★★★★	Media	Sí
Compacter++	2023	Adaptadores hiperbólicos	~1%	No	★★★★	★★★★	Alta	Sí

Fuente: Elaboración propia.

La siguiente figura presenta la cronología de aparición de las técnicas de ajuste fino seleccionadas como representativas del estado del arte para los métodos PEFT, especialmente aplicables en entornos con recursos computacionales limitados.

Figura 3. *Evolución temporal de las técnicas de ajuste fino seleccionadas*

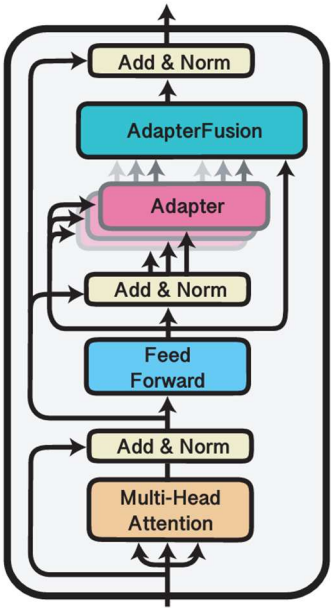


Fuente: Elaboración propia.

2.5.1. AdapterFusion (2020–2022)

AdapterFusion permite integrar múltiples adaptadores entrenados previamente en un único modelo base, facilitando aprendizaje multitarea mediante la selección dinámica de adaptadores relevantes para cada entrada, sin reentrenar el modelo completo (Pfeiffer et al., 2021). La arquitectura consiste en fusionar los vectores de salida de estos adaptadores usando un mecanismo de atención que decide qué adaptador es más relevante para una entrada determinada.

Figura 4. *Arquitectura de **AdapterFusion** integrada en un modelo Transformer, mostrando cómo se combinan múltiples adaptadores mediante atención ponderada*

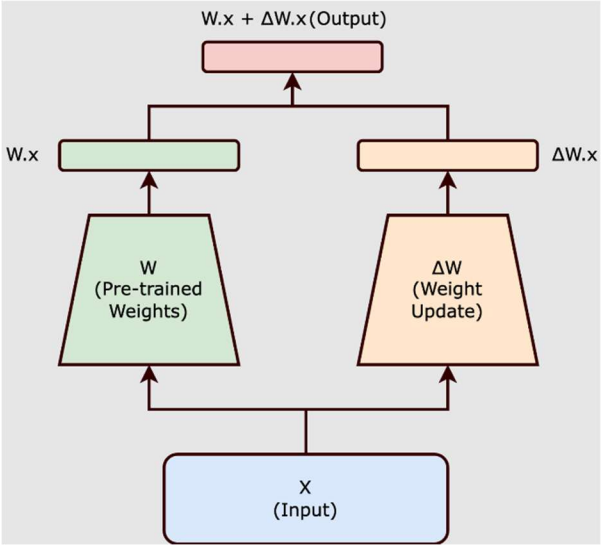


Fuente: (Pfeiffer et al., 2021).

2.5.2. LoRA (2021)

LoRA (Low-Rank Adaptation) introduce matrices entrenables de bajo rango paralelas a las capas lineales del modelo. Durante el entrenamiento, se actualizan únicamente estas matrices, mientras que los pesos del modelo base permanecen congelados, reduciendo significativamente el consumo de recursos durante el entrenamiento (Hu et al., 2021).

Figura 5. *Esquema del funcionamiento del método **LoRA**. Pesos congenlados (W) y pesos entrenables (ΔW)*

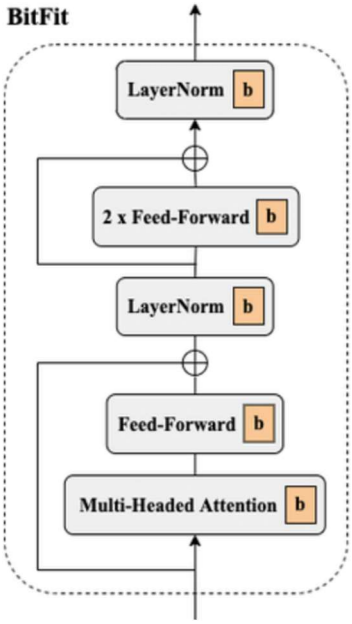


Fuente: (Jawade, 2023).

2.5.3. BitFit (2021)

BitFit es una técnica que ajusta únicamente los parámetros de sesgo (bias) de cada capa del modelo, representando una fracción mínima del total de parámetros, lo que resulta especialmente útil en tareas sencillas con recursos muy limitados (Ben Zaken et al., 2021).

Figura 6. *Esquema del funcionamiento del método **BitFit** integrado en un modelo transformer. Todos los bias son entrenables en las distintas capas del transformer*

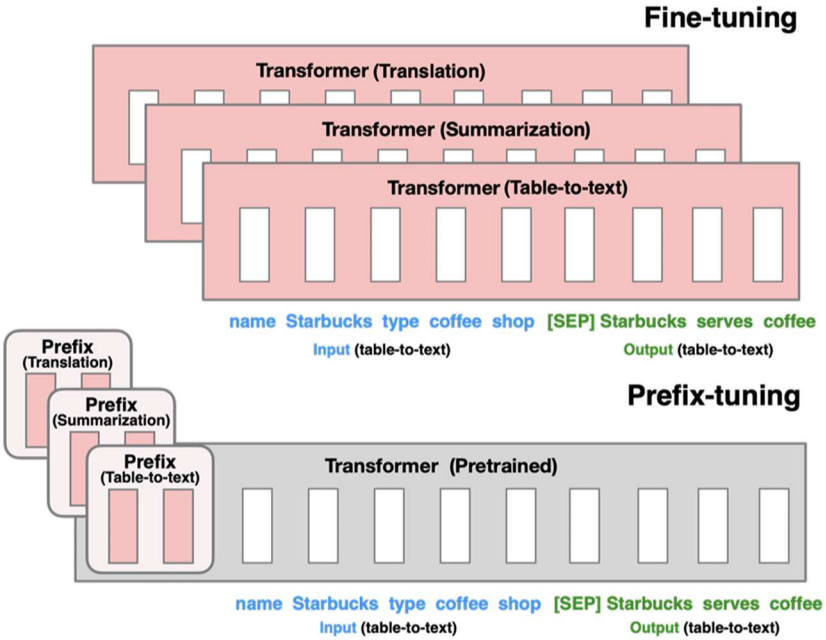


Fuente: (Fu et al., 2022).

2.5.4. Prefix Tuning (2021)

Prefix Tuning concatena secuencias de vectores entrenables al inicio de cada secuencia de entrada, actuando como tokens virtuales que modifican las representaciones internas del modelo e influyen en su atención, sin afectar sus pesos originales. Mientras que el ajuste fino clásico requiere actualizar todos los parámetros del modelo, *prefix tuning* congela el modelo base y solo entrena un conjunto pequeño de vectores llamados prefijos, haciendo el proceso modular y eficiente en memoria (Li & Liang, 2021).

Figura 7. *Comparación entre el **fine-tuning** tradicional (arriba) y **prefix tuning** (abajo)*

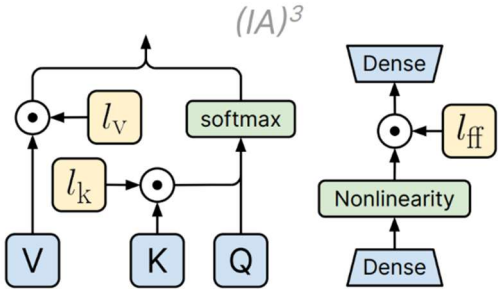


Fuente: (Li & Liang, 2021).

2.5.5. $(IA)^3$ (2022)

$(IA)^3$ (Infused Adapter by Inhibiting and Amplifying Inner Activations) introduce vectores aprendidos que reescalan claves, valores y activaciones internas en módulos específicos del modelo (atención y feedforward), logrando una adaptación eficiente sin alterar los pesos base del modelo, ideal para entornos restringidos computacionalmente (H. Liu et al., 2022) (Verma et al., 2024).

Figura 8. *Diagrama de **$(IA)^3$**. Se utilizan múltiples funciones de pérdida para asegurar un aprendizaje más rápido y eficaz del modelo*

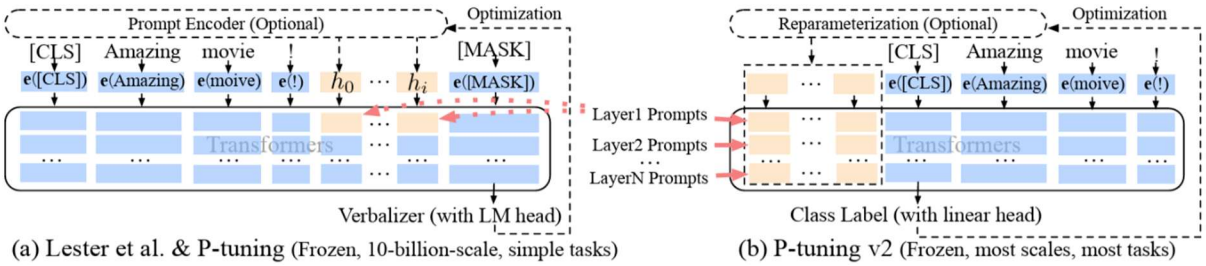


Fuente: (H. Liu et al., 2022).

2.5.6. P-Tuning v2 (2022)

P-Tuning v2 extiende la idea original de prompts entrenables insertando vectores en múltiples capas del modelo, permitiendo ajustes más profundos y mejorando sustancialmente la efectividad frente a versiones anteriores. El enfoque se basa en la modificación de embeddings internos mediante parámetros que actúan como claves semánticas (Weng, 2024).

Figura 9. ***P-Tuning v2*** *mejora la capacidad de adaptación respecto a P-Tuning insertando prompts en múltiples capas del modelo*



Fuente: (X. Liu et al., 2021b) .

2.5.7. AdaLoRA (2022)

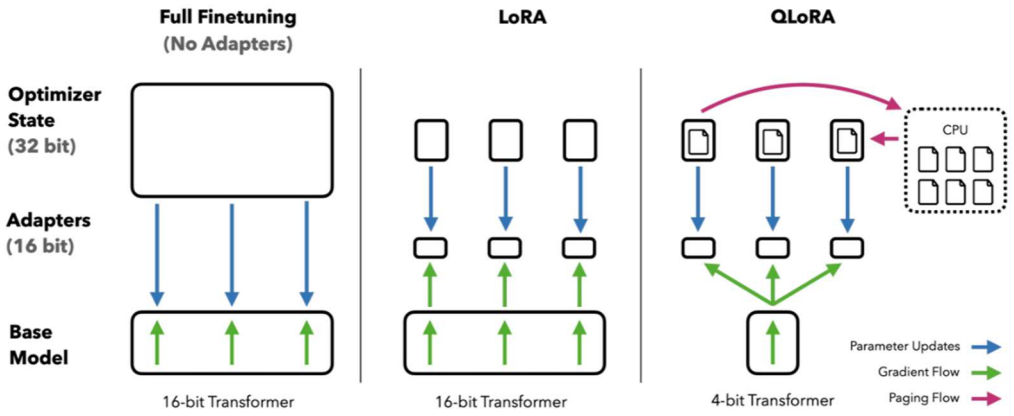
AdaLoRA mejora LoRA mediante la asignación dinámica del presupuesto de parámetros. Evalúa la importancia de cada matriz del modelo mediante técnicas como SVD (Singular Value Decomposition) y asigna más capacidad a las matrices más críticas. Esto mejora el rendimiento en tareas específicas y previene el sobreajuste en regiones poco relevantes del modelo (Zhang et al., 2023).

2.5.8. QLoRA (2023)

QLoRA combina cuantización a 4 bits del modelo base con LoRA para permitir el ajuste fino de LLMs grandes en hardware de consumo. La arquitectura cuantiza las matrices originales

usando técnicas como NF4 (NormalFloat 4-bit) y entrena matrices de bajo rango adicionales, preservando precisión mientras reduce drásticamente el uso de memoria (Dettmers et al., 2023).

Figura 10. *Comparación de métodos.* ***QLoRA*** *mejora respecto a* ***LoRA*** *al cuantizar el modelo Transformer a precisión de 4 bits y utilizar optimizadores con paginación para gestionar picos de memoria*



Fuente: (Dettmers et al., 2023).

2.5.9. ReLoRA (2023–2024)

ReLoRA introduce reinicializaciones periódicas de las matrices LoRA durante el entrenamiento. Estas matrices se fusionan con el modelo y luego se reinician para permitir que nuevas actualizaciones se acumulen desde un estado limpio. La arquitectura incorpora además un scheduler en forma de diente de sierra para la tasa de aprendizaje, lo que mejora la convergencia y estabilidad (Lialin et al., 2023).

2.5.10. Otras técnicas relevantes

Técnicas como MeZO, Delta Tuning y Compacter++ exploran enfoques alternativos de parametrización eficiente, como optimización de orden cero, almacenamiento de diferencias parametrizadas y descomposición hiperbólica, respectivamente (Zhu et al., 2023), (Lv et al., 2023).

2.6. Conclusiones

La evolución de las técnicas de fine-tuning eficiente en parámetros ha experimentado un crecimiento acelerado entre 2020 y 2024, motivada principalmente por la necesidad de facilitar el acceso a los modelos *LLM* sin depender de grandes infraestructuras de cómputo.

El desarrollo de estas técnicas representa un avance clave hacia la democratización del acceso a la inteligencia artificial avanzada, permitiendo que organizaciones con presupuestos y recursos limitados puedan beneficiarse del potencial de los modelos de lenguaje de gran tamaño.

2.6.1. Fortalezas detectadas

- Reducción significativa del número de parámetros entrenables.
- Preservación de los pesos base del modelo, garantizando privacidad de datos.
- Modularidad y reutilización en tareas múltiples.
- Rendimiento competitivo respecto al fine-tuning completo.

2.6.2. Debilidades o retos pendientes

- Algunas técnicas requieren calibración cuidadosa de hiperparámetros.
- Posible reducción de efectividad en modelos pequeños o con datos limitados.
- Falta de métodos estandarizados para comparar eficientemente técnicas entre sí.

2.6.3. Problemas abiertos

- Identificar qué técnica es más robusta y generalizable en un entorno de producción.
- Determinar combinaciones óptimas de técnicas según la tarea específica.
- Evaluar la escalabilidad y aplicabilidad en escenarios reales con recursos heterogéneos.

3. Objetivos y metodología de trabajo

Este capítulo establece la metodología de trabajo que guiará la evaluación comparativa de diferentes técnicas de fine-tuning eficiente en parámetros (PEFT) para modelos de lenguaje de gran tamaño (LLMs) en entornos con recursos computacionales limitados. Siguiendo esta metodología se pretende abordar dos objetivos principales:

- Evaluar el rendimiento de los modelos LLM en términos de uso de recursos hardware.
- Analizar la capacidad de las técnicas para realizar un fine-tuning efectivo en tareas de traducción de lenguaje natural a lenguaje SQL.

3.1. Objetivo general

El objetivo general del presente trabajo es evaluar comparativamente la efectividad, eficiencia computacional y aplicabilidad práctica de diferentes técnicas de ajuste fino eficiente en parámetros para modelos de lenguaje de gran tamaño, en contextos con recursos computacionales limitados.

La comparativa se enfocará en un subconjunto de técnicas PEFT, en concreto: ***LoRA***, ***QLoRA***, ***BitFit***, ***Prefix Tuning v2*** e ***(IA)³***. Se emplearán métricas objetivas de rendimiento tales como la precisión, el consumo de memoria y el tiempo de inferencia, obtenidas a partir de la evaluación de benchmarks estándar y herramientas de evaluación aplicadas sobre modelos LLM preentrenados. Esta comparativa pretende aportar valor práctico significativo para escenarios reales de aplicación en entornos de producción industrial y empresarial.

3.2. Objetivos específicos

Para lograr el objetivo general planteado para este trabajo, se establecen los siguientes objetivos específicos:

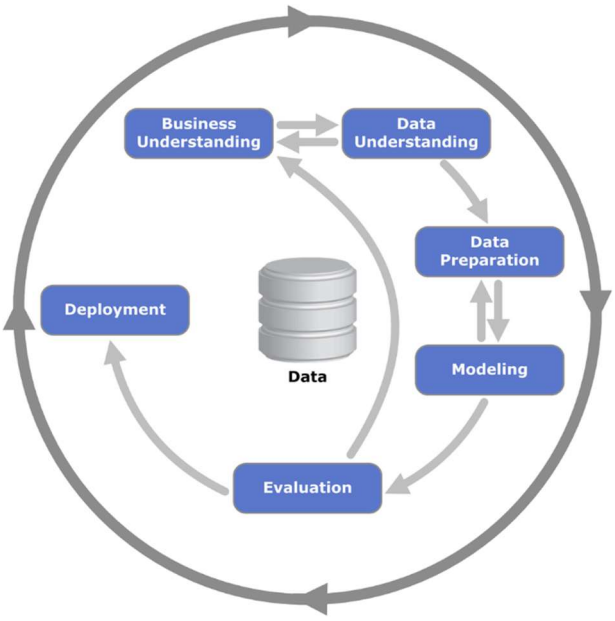
- **Analizar críticamente** el estado del arte actual de técnicas PEFT aplicadas a LLMs para identificar características diferenciales y contextos de uso recomendados.
- **Seleccionar un conjunto representativo** de técnicas PEFT en base a criterios documentados como: precisión, eficiencia computacional, sobrecarga en inferencia, complejidad y modularidad.

- **Seleccionar los modelos LLM** más adecuados para realizar el fine-tuning usando las técnicas PEFT seleccionadas. Para ello, se elegirá uno o varios de los siguientes modelos preentrenados, teniendo en cuenta factores como el tamaño del modelo, la eficiencia computacional y la capacidad de adaptarse a tareas de procesamiento del lenguaje natural: GPT-2, DistilBERT, ALBERT, TinyBERT, ELECTRA, T5, LLaMA (versiones pequeñas: LLaMA-7B, LLaMA-13B), etc.
- **Implementar experimentalmente** cada técnica seleccionada sobre on modelo preentrenado común para garantizar consistencia en la comparativa.
- **Evaluar cuantitativamente** el rendimiento y eficiencia computacional de cada técnica utilizando benchmarks sobre una misma plataforma con la misma configuración de recursos hardware y software.
- **Sintetizar los resultados obtenidos** mediante tablas comparativas y gráficos ilustrativos que faciliten una interpretación objetiva y precisa de los mismos.

3.3. Metodología de trabajo

CRISP-DM es la metodología estándar de minería de datos que se aplicará en este proyecto de comparación de diferentes enfoques de fine-tuning de modelos LLM. A continuación, se detalla cómo se adaptará cada fase de CRISP-DM para llevar a cabo esta comparativa:

Figura 11. *Diagrama de proceso. Relación entre las diferentes fases de CRISP-DM*



Fuente: (Shearer, 2000).

3.3.1. Definición del problema y objetivos de negocio (Business Understanding)

Evaluar comparativamente la efectividad, eficiencia computacional y aplicabilidad práctica de diferentes técnicas de fine-tuning para modelos LLM en escenarios con recursos computacionales limitados formulando los objetivos SMART: (Específico, Medible, Alcanzable, Relevante, Tiempo).

3.3.2. Comprensión del dominio y los datos (Data Understanding)

Obtener y explorar los datos (modelos preentrenados y datasets de pruebas) para comprender sus características, calidad y cómo se pueden usar en el fine-tuning.

- **Explorar los modelos LLM seleccionados**: Obtener los modelos preentrenados de diferentes fuentes (por ejemplo, GPT-2 o LLaMA) y evaluar su estructura y tamaño.
- **Características de los modelos**: Investigar los recursos computacionales necesarios para entrenar los modelos preentrenados, tales como la memoria RAM, la GPU utilizada y el tiempo estimado para el fine-tuning.
- **Evaluación de los datos de entrada**: Recopilar datasets que servirán para evaluar las técnicas de fine-tuning. Esto incluye datasets de **traducción de lenguaje natural a SQL** o cualquier otra tarea relevante, como clasificación de texto o generación de preguntas.
- **Análisis de calidad de los datos**: Revisar si los datos de entrada contienen valores nulos, ruidos o inconsistencias que puedan afectar la efectividad del fine-tuning.

3.3.3. Preparación de los datos (Data Preparation)

Preprocesar los datos y configurar los modelos para que estén listos para el fine-tuning con las técnicas seleccionadas.

- **Selección de las Métricas de Evaluación**: Para realizar una comparativa precisa entre las técnicas, se deben definir métricas objetivas y subjetivas que permitan evaluar tanto la eficiencia como la efectividad de cada técnica:
    - **Para el Objetivo 1 (hardware con bajos recursos)***:*

        - **Uso de memoria:** Medir la cantidad de memoria RAM utilizada durante el proceso de fine-tuning.
        - **Uso de CPU/GPU**: Medir la carga de la CPU y la GPU durante el proceso de fine-tuning.
        - **Tiempo de entrenamiento**: Contabilizar el tiempo necesario para completar el fine-tuning de cada técnica.
    - **Para el Objetivo 2 (traducción de lenguaje natural a SQL)***:*
        - **Exactitud literal (Exact Match):** Medir la calidad de las traducciones generadas, evaluando si las consultas SQL son correctas y optimizadas.
        - **Tiempo de inferencia**: Evaluar el tiempo que toma generar una consulta SQL a partir de una frase en lenguaje natural.
- **Preparación del entorno de prueba**:
    - Utilizar un entorno con recursos limitados (por ejemplo, una GPU de bajo rendimiento o una CPU con memoria limitada) para probar las técnicas en el primer objetivo.
    - Asegurar que todas las técnicas seleccionadas se aplican sobre el mismo modelo LLM base para así evitar sesgos.
- **Preprocesamiento de datos**:
    - Para el primer objetivo, recopilar un conjunto de datos de benchmark con diferentes tareas de fine-tuning, asegurando que el proceso se pueda realizar dentro de los límites temporales y de hardware.
    - Para el segundo objetivo, seleccionar un conjunto de datos de preguntas en lenguaje natural relacionadas con SQL, y el modelo de datos asociado para realizar la traducción. Esto podría incluir preguntas complejas que requieran una representación de datos avanzada.

3.3.4. Modelado (Modeling)

Aplicar las diferentes técnicas de fine-tuning a los modelos seleccionados y entrenarlos para obtener los mejores resultados posibles.

- **Implementación de las técnicas**:

  - Implementar cada una de las técnicas de fine-tuning en el mismo conjunto de datos.
  - Asegurar que el ajuste de parámetros sea coherente en todas las técnicas (por ejemplo, el número de parámetros entrenables debe ser el mismo en cada técnica).
- **Prueba en un entorno con pocos recursos hardware (Objetivo 1)**:
  - Ejecutar el proceso de fine-tuning para cada técnica y medir las métricas relacionadas con el uso de recursos: la memoria, la carga de CPU/GPU y el tiempo de entrenamiento.
  - Realizar pruebas de escalabilidad variando los recursos disponibles (por ejemplo, reduciendo la memoria o la potencia de la GPU) y observando cómo se comportan las técnicas.
- **Prueba de traducción de lenguaje natural a SQL (Objetivo 2)**:
  - Entrenar el modelo ajustado para cada técnica, utilizando el mismo conjunto de datos de traducción de lenguaje natural a SQL.

3.3.5. Evaluación (Evaluation)

Evaluar, mediante las métricas definidas, el rendimiento de cada modelo ya ajustado.

- **Comparativa de uso de recursos (Objetivo 1)**:
  - Para cada una de las técnicas, evaluar el uso de memoria, tiempo de entrenamiento y otros recursos, mediante la creación de gráficos que permitan comparar y analizar su desempeño bajo las mismas condiciones.
  - Determinar cuál de las técnicas es más eficiente en términos de uso de recursos y escalabilidad en hardware con bajos recursos.
- **Comparativa de rendimiento en la tarea de traducción a SQL (Objetivo 2)**:
  - Analizar las traducciones generadas por cada técnica utilizando las métricas de precisión, tiempo de inferencia y tasa de éxito.
  - Determinar qué técnica produce las consultas SQL más correctas y eficientes en términos de ejecución.

3.3.6. Despliegue (Deployment)

En el contexto de este proyecto, la etapa de Despliegue está fuera de su alcance, ya que ninguno de los modelos ajustados se ejecutará dentro de un entorno de producción. Por tanto, el foco principal de este trabajo es hacer una comparación de las metodologías de fine-tuning de modelos LLM para analizar cuál de ellas es la mejor para su aplicación en una infraestructura con recursos computacionales limitados.

Únicamente se estudiarán y evaluarán los métodos de fine-tuning bajo condiciones controladas, midiendo su eficiencia computacional y su desempeño en tareas concretas como la traducción del lenguaje natural a SQL. Se deja fuera la implementación y el despliegue de los modelos adaptados, ya que el fin principal es la entrega de una recomendación respecto a cuál de las técnicas es la más indicada para aplicar en entornos con hardware restringido, pero de manera tal que esto no se asocie a su despliegue en un sistema real o de pruebas.

Se limitan, por tanto, todas las conclusiones y recomendaciones que se desprendan de este estudio a las etapas de evaluación y comparación, y no se extienden hacia la aplicación práctica de los modelos dentro de un entorno operativo.

4. Planteamiento de la comparativa

Este trabajo pretende efectuar un estudio detallado de varias técnicas de **fine-tuning eficiente de parámetros (PEFT)** (Brown et al., 2020) (Howard & Ruder, 2018) al aplicarlas a **modelos de lenguaje de gran tamaño (LLM)** (Naveed et al., 2023)(Minaee et al., 2024), para establecer la que mejor se ajusta para su uso para entornos con hardware limitado. Debido a la creciente necesidad del uso de modelos LLM en aplicaciones de **procesamiento del lenguaje natural (NLP)**, se debe asegurar la mejora en su desempeño, pero sin sacrificar los recursos de computación disponibles, especialmente en entornos con limitaciones de memoria o potencia de procesado.

La comparación evalua cinco técnicas de fine-tuning bien descritas en la literatura especializada: **LoRA**, **QLoRA**, **BitFit**, **Prefix Tuning v2** y **(IA)³**. Cada una de estas técnicas tiene una forma diferente de abordar el ajuste fino de los modelos LLM, tratando de ajustar eficientemente tan solo una pequeña fracción de sus parámetros, lo que permite reducir muy notablemente los requerimientos de recursos computacionales, en contraposición a los métodos de fine-tuning tradicionales.

A través de esta comparación, se analizan las técnicas basándose en unos criterios determinados que abarcan el uso de memoria, el tiempo de inferencia, el desempeño para trabajos de traducción de lenguaje natural a SQL (Zhong et al., 2017a) y la eficiencia con recursos hardware limitados. Este análisis pretende brindar directrices claras acerca de cuál de estas técnicas es la mejor para su implementación en entornos con infraestructura de recursos restringidos, sin afectar la calidad o la precisión de las aplicaciones que se implementen.

El desarrollo de esta comparativa se basará en la metodología CRISP-DM, que ofrece un enfoque estructurado, aunque flexible para proyectos de inteligencia artificial. Ya que CRISP-DM permite hacer un análisis sistemático desde la comprensión del negocio y los datos hasta la evaluación de los resultados. Este acercamiento asegurará que se lleve a cabo una evaluación sólida y objetiva de cada una de las técnicas de fine-tuning, lo que permitirá sacar conclusiones claras, fundamentadas en datos, acerca de la mejor opción para optimizar el desempeño de los modelos LLM dentro de entornos con limitaciones de recursos.

4.1. Definición del problema y objetivos de negocio (Business Understanding)

El crecimiento del desarrollo de interfaces que facilitan la traducción del **lenguaje natural a consultas SQL sobre los datos (NL-to-SQL)** (Quamar et al., 2022) es un dominio de gran interés dentro del contexto de la inteligencia artificial aplicada a la interacción con bases de datos. Tales interfaces pretenden hacer disponible, a los usuarios finales, el acceso a la información estructurada almacenada en bases de datos relacionales, sin necesidad de poseer conocimientos avanzados de SQL. La posibilidad de comprender preguntas expresadas utilizando lenguaje natural y su correspondiente consulta SQL se ha convertido en un problema crítico para el desarrollo de sistemas inteligentes, especialmente para campos como el análisis de negocio, la enseñanza o la atención al cliente.

- **Formulación de objetivos SMART**:
    - **Específico**: Determinar en base a la comparativa entre técnicas de fine-tuning cuál de ellas es la que arroja mejores resultados en la tarea de traducción de lenguaje natural a SQL optimizando el uso de recursos computacionales disponibles en el entorno de ejecución.
    - **Medible**: Evaluar tanto la eficiencia computacional (en términos de uso de recursos) como la tarea de NL-to-SQL:
        - **Objetivo 1:** Evaluar el rendimiento de las técnicas en hardware con bajos recursos. Se busca identificar cuál de entre las técnicas de fine-tuning seleccionadas puede adaptarse mejor a un entorno de hardware con recursos limitados (por ejemplo, en términos de memoria RAM y capacidad de procesamiento de CPU/GPU).
        - **Objetivo 2**: Evaluar la capacidad de cada técnica para realizar un fine-tuning efectivo que mejore el desempeño del modelo en la tarea NL-to-SQL en base a un modelo de datos específico.
    - **Alcanzable**: Verificar que las técnicas seleccionadas (LoRA, QLoRA, BitFit, Prefix Tuning v2 e $(IA)^3$) se pueden implementar con el hardware disponible.
    - **Relevante**: El objetivo debe tener un impacto claro en la mejora del tiempo de entrenamiento de los modelos, para empresas que no disponen de grandes recursos computacionales.

- **Tiempo**: Completar la evaluación de las técnicas de fine-tuning en un plazo de 6 semanas, con un TFM final sobre el rendimiento de cada técnica listo para ser presentado.

4.2. Comprensión del dominio y los datos (Data Understanding)

4.2.1. Análisis de los modelos LLM seleccionados

En este apartado se evalúan algunos modelos LLM para establecer cuál de ellos se ajusta mejor para la aplicación de técnicas de fine-tuning en entornos con recursos limitados. La comparativa se ha ajustado a varios parámetros clave que nos facilitan la posibilidad de comparar tanto la eficiencia computacional como la adaptabilidad a tareas particulares; como la traducción de lenguaje natural a SQL.

4.2.1.1. Características consideradas en la comparativa

A continuación, se detallan las principales características que se han tenido en cuenta para realizar la comparativa de modelos:

- **Recursos hardware necesarios para aplicar técnicas de fine-tuning**:
  - Evaluación de la **memoria** y de la **potencia de procesamiento** (CPU/GPU) requeridas para realizar el fine-tuning de cada modelo.
  - Estudio de cómo cada modelo maneja los recursos computacionales en **entornos con hardware limitado**.
- **Antigüedad del modelo**:
  - El año de lanzamiento de cada modelo es relevante para comprender su madurez y si ha sido optimizado para tareas más recientes.
- **Popularidad**:
  - Popularidad dentro de la comunidad de la industria y la investigación, que se refleja en el uso y adopción general del modelo. Modelos populares tienen más documentación, mayor número de citas académicas, ejemplos de uso y soporte.
- **Adecuación para la traducción de lenguaje natural a SQL**:

- Análisis de la capacidad de cada modelo para la **generación de consultas SQL** a partir de entradas en **lenguaje natural,** una tarea crucial para muchos sistemas de bases de datos y aplicaciones de IA.

4.2.1.2. Comparativa de modelos LLM

En la siguiente sección, se comparan detalladamente diferentes modelos LLM, considerando un conjunto de características clave que resultan importantes para juzgar su desempeño y conformidad dentro de un entorno de fine-tuning con recursos limitados.

Tabla 3. *Comparativa de modelos LLM en términos de recursos computacionales, popularidad y aplicabilidad para la tarea de traducción de lenguaje natural a SQL*

Técnica PEFT	Año aparición	Popularidad[1]	Eficiencia computacional	NL-to-SQL
DistilBERT (Sanh et al., 2019)	2019	★★★	★★★★★ Modelo ligero y eficiente. Su arquitectura es más pequeña que BERT y mantiene un rendimiento similar. Fácil de fine-tunear en entornos limitados.	★★★ Aunque es eficiente, su capacidad para tareas NL-SQL es limitada debido a la pérdida de información semántica respecto a BERT. PEFT mejora algo el rendimiento, pero sigue siendo inferior a modelos más robustos.
ALBERT (Lan et al., 2019)	2019	★★★	★★★★ Optimización sobre BERT, reduciendo parámetros mediante factorización y compartición de pesos. Es eficiente, aunque no tanto como DistilBERT o TinyBERT.	★★★★ Buena comprensión semántica y optimización de parámetros. Con PEFT, su capacidad para manejar NL-SQL puede ser destacable, aunque no tanto como modelos generativos.
T5 (Text-to-Text Transfer Transformer) (Raffel et al., 2020)	2019	★★★★	★★★ Los modelos T5 son bastante grandes, aunque las versiones pequeñas (T5-Small) pueden ser más manejables.	★★★★★ T5 está diseñado para tareas de secuencia a secuencia, lo cual incluye traducción de texto a SQL. PEFT maximiza su rendimiento sin sacrificar precisión.

[1] La métrica de popularidad se ha estimado a partir del número de citas académicas, la presencia en librerías de código abierto, y su adopción práctica en benchmarks y proyectos reconocidos. Fuente: elaboración propia basada en Hugging Face, Google Scholar y GitHub.

BART **(M. Lewis et al., 2020)**	2019	★★★	★★ Aunque optimizado para generación de texto, su arquitectura auto-regresiva requiere más capacidad de cómputo para el fine-tuning.	★★★★ Muy efectivo en tareas de generación de texto estructurado, como NL-SQL. PEFT permite un ajuste fino sin necesidad de entrenar el modelo completo.
GPT-2 **(Radford et al., 2019)**	2019	★★★★	★★★ Menos costoso que LLaMA-7B, pero aún requiere un hardware significativo para ajustar adecuadamente sus parámetros.	★★★★ Capaz de generar SQL de manera efectiva. PEFT permite afinar su conocimiento, aunque es menos estructurado que T5 o BART para esta tarea específica.
TinyBERT **(Jiao et al., 2020)**	2020	★★★	★★★★★ Incluso más pequeño que DistilBERT. Está optimizado para eficiencia máxima en tareas específicas. Ideal para dispositivos de baja capacidad.	★★ Muy ligero, pero sacrifica contexto y comprensión profunda, lo cual es crítico en la conversión NL-SQL. PEFT no logra suplir estas limitaciones.
MobileBERT **(Sun et al., 2020)**	2020	★★★	★★★★★ Diseñado específicamente para entornos móviles. Muy ligero y eficiente para el fine-tuning, con un buen equilibrio entre rendimiento y uso de recursos.	★★★ Mejor comprensión semántica que TinyBERT, pero sigue siendo limitado para tareas NL-SQL complejas. PEFT ayuda, pero no llega al nivel de modelos más robustos.
Reformer **(Kitaev et al., 2020)**	2020	★★	★★ Utiliza técnicas de atención local para ser más eficiente, pero todavía puede ser demandante.	★★ Adecuado para generación de texto y tareas de traducción, aunque no es específico para SQL.
ELECTRA **(Clark et al., 2020)**	2020	★★★	★★★★ Muy eficiente, pero el fine-tuning requiere un poco más de recursos que DistilBERT. Aun así, es bastante optimizado.	★★★★ Su mecanismo de preentrenamiento basado en discriminación lo hace efectivo para identificar patrones en texto, lo cual es útil para NL-SQL. PEFT optimiza bien este modelo.
LLaMA-7B **(Touvron et al., 2023)**	2023	★★★	★★★ Muy potente, pero 7B de parámetros demandan un entorno computacional considerable para fine-tuning, incluso optimizado.	★★★★ Aunque es poderoso, su optimización para NL-SQL depende del fine-tuning. PEFT le permite un ajuste eficiente, pero necesita un entrenamiento adecuado para obtener buenos resultados.

Fuente: Elaboración propia.

4.2.1.3. Resumen y observaciones

1. **Modelos con requerimientos de hardware bajos**:
    - DistilBERT, TinyBERT y MobileBERT resultan los modelos más idóneos para entornos con pocos recursos. Resultan muy útiles para tareas menos complejas del NLP, como clasificación y análisis de opiniones, pero potencialmente requerirían ajustes si se aplicaran a tareas más complicadas como la traducción del lenguaje natural a SQL.
2. **Modelos eficientes**:
    - ALBERT y ELECTRA son opciones eficientes en términos de recursos computacionales. Ambos modelos funcionan bien para tareas de **NLP**. ELECTRA es especialmente adecuado para tareas de generación de texto y traducción de SQL.
3. **Modelos para tareas complejas**:
    - T5 y BART son recomendables para tareas de traducción de lenguaje natural a SQL, ya que están diseñados para tareas de transformación de texto a texto. Estos modelos son más pesados en términos de recursos computacionales, pero su capacidad para manejar tareas complejas de generación de texto los hace ideales para esta aplicación.
4. **Modelos de última generación**:
    - Los modelos LLaMA pueden proporcionar un rendimiento excelente, pero debido a su tamaño, requieren hardware potente para ser utilizados eficientemente en fine-tuning. LLaMA-7B es más adecuado para hardware limitado.
5. **Popularidad**:
    - Modelos como GPT-2 y T5 son muy populares, con una gran cantidad de ejemplos y documentación disponible. LLaMA y ELECTRA también están ganando popularidad debido a sus características de eficiencia y flexibilidad.

4.2.1.4. Conclusión

Después de la comparación de varios modelos de LLM y teniendo en consideración los fines de este proyecto, optamos por utilizar las versiones LLaMA-7B y GPT-2 para fine-tuning.

Debido a una conjunción de factores, facilidad de uso, habilidad para producir textos complejos y sus bajas demandas de recursos. A continuación, se argumenta esta elección.

- **Capacidad para generar textos complejos**:
    - GPT-2 es conocido por su capacidad para producir textos de alta calidad, lo que lo hace adecuado para tareas complejas como la traducción de lenguaje natural a SQL. Su arquitectura es capaz de generar textos coherentes y contextualmente apropiados, una habilidad que se aplica a la generación de consultas SQL a partir de texto en lenguaje natural.
    - LLaMA-7B, se ha revelado con un buen desempeño en tareas de NLP complejas. Su tamaño de 7 mil millones de parámetros permite realizar tareas complejas de generación de texto con menos requisitos hardware en comparación con sus versiones grandes como LLaMA-13B.
- **Requerimientos de recursos moderados**:
    - GPT-2 se ofrece en tamaños más pequeños (como GPT-2-124M), que ocupan menos memoria y tienen una necesidad computacional más reducida, lo que facilita el fine-tuning para sistemas con pocos recursos. Su desempeño para la generación de texto y su adaptabilidad para tareas específicas como la traducción del lenguaje natural a SQL es muy buena a medida que se ajusta a un entorno de recursos bajas.
    - LLaMA-7B, por otro lado, es una opción más ligera dentro de la familia LLaMA, que, aunque proporciona un alto nivel de capacidad para producir textos complejos, mantiene un equilibrio óptimo entre el rendimiento y las demandas de hardware. Su tamaño más pequeño, en comparación con modelos como LLaMA-13B, lo que lo hace adecuado para ejecutarse de manera eficiente con hardware limitado, sin perder competitividad en su desempeño.
- **Popularidad y documentación abundante**:
    - GPT-2 y LLaMA-7B son modelos ampliamente conocidos y muy populares en la comunidad de investigación y desarrollo, lo que garantiza una gran disponibilidad de documentación, ejemplos de fine-tuning y otros recursos. Esta popularidad no solo asegura una fácil implementación y adaptación de los

métodos, sino que también ofrece una sólida base de conocimientos y soporte, algo esencial para garantizar un fine-tuning exitoso.

En resumen, GPT-2 y LLaMA-7B son los mejores candidatos para este proyecto por su capacidad para realizar tareas complejas de texto, como la conversión de lenguaje natural a SQL, sin sacrificar el desempeño en entornos de hardware limitado.

4.2.2. Características de los modelos

En esta sección se van a detallar las características de los modelos GPT-2 (Radford et al., 2019) y LLaMA-7B (Touvron et al., 2023) para analizar sus capacidades y flexibilidad para el fine-tuning en entornos de recursos restringidos. La comparación se efectuará por varios parámetros clave que nos brindarán la oportunidad de destacar con claridad las características técnicas y del desempeño de cada modelo. A lo largo de los siguientes apartados, se pretende brindar una percepción clara de cómo se comporta cada modelo al realizar tareas complejas de procesamiento de lenguaje natural (NLP), y en particular cuando se trata de aplicaciones de traducción de lenguaje natural a SQL.

Los parámetros de evaluación incluyen:

- **Nombre del modelo**: Se especificará el nombre de cada modelo para identificar claramente las versiones evaluadas.
- **Familia**: Familia a la que pertenece el modelo LLM.
- **Año de aparición**: Se tomará en cuenta el año de lanzamiento de cada modelo para entender su nivel de madurez.
- **Arquitectura del modelo**: Se describirá la arquitectura de cada modelo, para entender mejor cómo procesan y generan texto.
- **Tamaño del modelo (Parámetros)**: Se evaluará el número de parámetros de cada modelo, lo cual influye directamente en su capacidad de generalización, su precisión en tareas complejas y los recursos computacionales que requiere su ajuste.
- **Recursos hardware necesarios**: Se analizarán los requisitos de CPU/GPU y memoria RAM necesarios para ejecutar y entrenar el modelo.

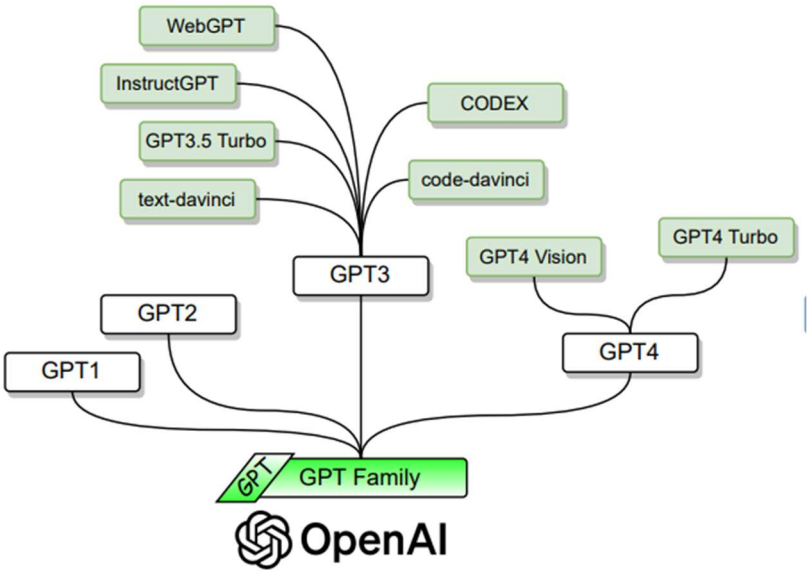
- **Tareas de NLP que soporta**: Se detallarán las tareas específicas de procesamiento de lenguaje natural para las que cada modelo está diseñado o ha demostrado ser efectivo, como generación de texto, clasificación de texto, y traducción de lenguaje natural a SQL.
- **Métricas de Desempeño**: Se evaluará el rendimiento de cada modelo en función de métricas clave como *precisión*, *velocidad de inferencia*, y *consumo de recursos*, lo que permitirá comparar su eficiencia en diferentes entornos de ejecución.
- **Ventajas y limitaciones**: Se identificará lo que distingue a cada modelo, así como sus posibles limitaciones, especialmente en cuanto a *requerimientos de recursos* y *capacidad para tareas específicas como la traducción de SQL*.

A través de esta evaluación, se pretende determinar qué modelo, entre GPT-2 y LLaMA-7B, es más adecuado para realizar el fine-tuning en entornos con recursos limitados, asegurando un equilibrio entre rendimiento, precisión y eficiencia.

4.2.2.1. Características del modelo GPT-2

- **Nombre del modelo:** GPT-2 Small (Generative Pre-trained Transformer 2)
- **Familia**: GPT, publicada por OpenAI.

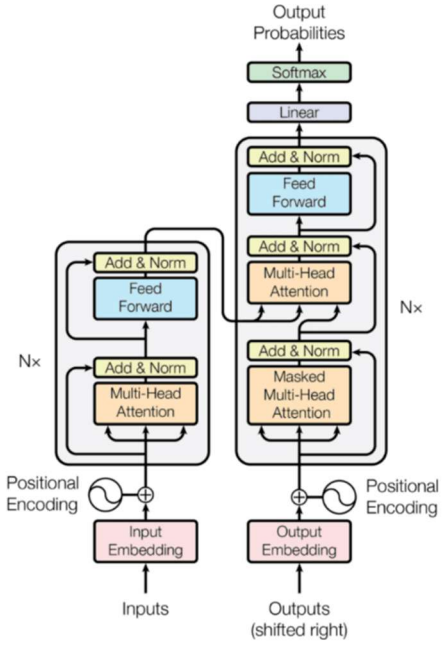
Figura 12. *Familia GPT*



Fuente: (Minaee et al., 2024)

- **Año de aparición**: 2019
- **Arquitectura del modelo**:
    - **Transformers**: GPT-2 está construido sobre la arquitectura transformer, que utiliza mecanismos de atención para aprender representaciones del lenguaje de manera no secuencial. Esta arquitectura permite al modelo generar texto coherente y de alta calidad a partir de una cantidad limitada de texto inicial.

Figura 13. *Arquitectura Transformers*



Fuente: (Vaswani et al., 2017).

- **Tamaño del modelo**:
    - 124 millones de parámetros (en la versión Small).
    - También existen versiones mayores con **355M**, **774M** y **1.5B** parámetros.
- **Recursos hardware necesarios**:
    - **GPU**: La versión de 124 millones de parámetros de GPT-2 puede ejecutarse en GPUs con tan solo **4–8 GB de VRAM.** No se requiere una GPU de gama alta para la inferencia, y se puede incluso ejecutar en CPU modernas para tareas ligeras, aunque con mayor latencia.
    - **Memoria**: El fine-tuning básico del modelo GPT-2-124M puede realizarse con **8–16 GB de RAM**, especialmente si se usan técnicas como gradient accumulation o entrenamiento en batches pequeños.

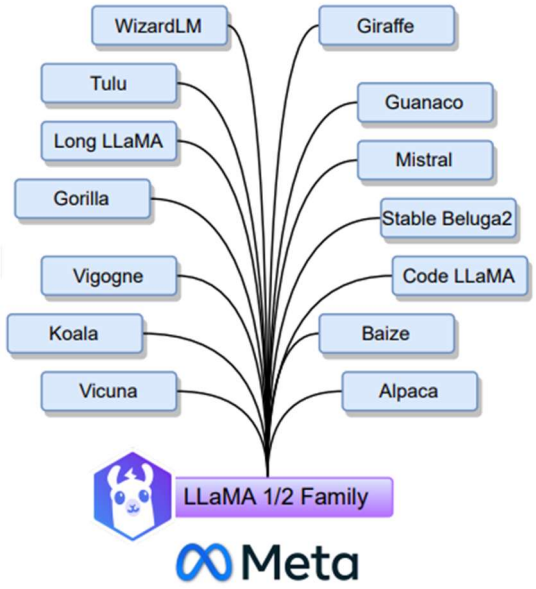
- **Popularidad y uso en la comunidad**:
    - **Alta**: GPT-2 es uno de los modelos de LLM más conocidos y utilizados en la comunidad de NLP. Ha sido ampliamente adoptado tanto en la investigación como en la industria, con numerosas implementaciones y tutoriales disponibles en plataformas como Hugging Face y GitHub.
- **Tareas de NLP que soporta**:
    - **Generación de texto**: GPT-2 es excepcionalmente bueno generando texto coherente, lo que lo hace adecuado para completado de texto, generación de historias, resúmenes, y respuestas automáticas.
    - **Traducción de lenguaje natural**: Con el ajuste correcto, GPT-2 puede hacer la tarea de conversión de lenguaje natural a SQL, ajustándose a las conversiones de lenguaje natural a otros tipos de formatos estructurados como SQL.
- **Métricas de desempeño**:
    - **Precisión**: Dependiendo de la tarea y del fine-tuning realizado, GPT-2 puede alcanzar una precisión muy alta.
    - **Velocidad de inferencia**: La velocidad varía según el tamaño del modelo. La versión más pequeña de GPT-2 tiene tiempos de inferencia más rápidos, lo que lo hace más adecuado para entornos con menos recursos.
    - **Consumo de memoria**: Un modelo pequeño como GPT-2-124M es más ligero en términos de memoria, mientras que la versión más grande de GPT-2 (1.5B parámetros) es mucho más demandante en recursos.
- **Ventajas y limitaciones**:
    - **Ventajas**:
        - **Generación de texto coherente**: GPT-2 es muy competente para generar texto coherente y fluido, lo que lo hace apropiado para tareas creativas y de conversación.
        - **Versatilidad**: Aunque originalmente diseñado para la generación de texto, se adapta bien a diversas tareas de NLP mediante fine-tuning.
    - **Limitaciones**:
        - **Requerimientos de recursos**: Las versiones grandes (como GPT-2-1.5B;) pueden requerir hardware potente, lo que hace que no sea tan

adecuado para entornos con recursos extremadamente limitados sin optimizaciones.

4.2.2.2. Características del modelo LLaMA-7B

- **Nombre del modelo**: LLaMA-7B (Large Language Model Meta AI, versión de 7 mil millones de parámetros)
- **Familia**: LLaMA 1/2, publicada por Meta.

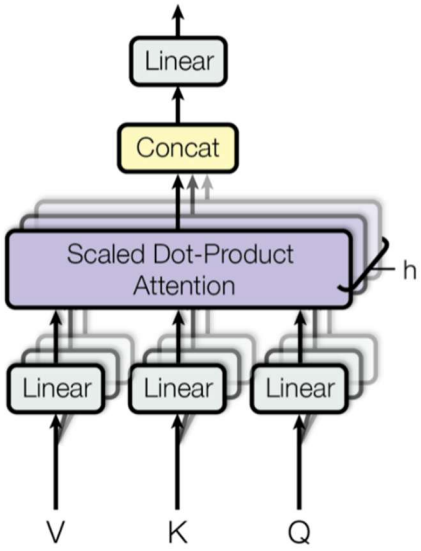
Figura 14. *Familia LLaMA 1/2*



Fuente: (Minaee et al., 2024)

- ***Año de aparición***: 2023
- ***Arquitectura del modelo***:
  - ***Transformers***: LLaMA-7B se basa en la arquitectura de transformers desarrollada por Meta, similar a la utilizada en modelos anteriores como GPT y BERT. Utiliza el mecanismo de *multi-head attention* (Vaswani et al., 2017) para procesar el texto de manera eficiente, permitiendo un entrenamiento rápido y una buena capacidad de generación de texto.

Figura 15. *Múltiples capas de atención ejecutadas en paralelo, donde V, K y Q representan respectivamente las matrices de values, keys y queries*



Fuente: (Vaswani et al., 2017)

- ***Tamaño del modelo***:
  - ***7 mil millones de parámetros*** (LLaMA-7B), lo que lo convierte en la versión más ligera dentro de la familia de modelos LLaMA, por debajo de variantes como LLaMA-13B, LLaMA-30B o LLaMA-65B. Su menor tamaño permite un equilibrio razonable entre rendimiento y coste computacional.
- ***Recursos hardware necesarios***:
  - **GPU**: LLaMA-7B, siendo la versión más pequeña de la familia original de modelos LLaMA, presenta un perfil más accesible en términos de requerimientos de hardware. Para tareas de fine-tuning, se recomienda el uso de una GPU con al menos ***16 GB de VRAM***, aunque puede ejecutarse en entornos con recursos más modestos si se aplican técnicas como *gradient checkpointing* o *quantization*. Para la inferencia, es posible ejecutarlo incluso en CPU modernas, aunque con menor rendimiento y mayor latencia.
  - **Memoria**: Para un entrenamiento eficiente, se requiere ***16-32 GB de RAM***. Las versiones más pequeñas de LLaMA (como LLaMA-7B) permiten una mayor accesibilidad en términos de recursos en comparación con modelos más grandes.
- **Popularidad y uso en la comunidad**:

  - **Alta**: Aunque es relativamente reciente, LLaMA-7B ha ganado popularidad rápidamente dentro de la comunidad de investigación y en aplicaciones industriales. Meta ha hecho públicos varios modelos LLaMA, y la comunidad de Hugging Face y otras plataformas ya han implementado el modelo, proporcionando recursos y documentación.

- **Tareas de NLP que soporta**:
  - **Generación de texto**: LLaMA-7B es excelente para la generación de texto coherente y puede ser utilizado en una variedad de tareas como completado de texto, generación de resúmenes, y más.
  - **Clasificación de texto**: Como modelo de lenguaje generalista, puede ser ajustado para tareas de clasificación de texto y análisis de sentimientos.
  - **Traducción de lenguaje natural**: LLaMA-7B es útil para tareas como la generación de texto de alta calidad a partir de entradas estructuradas.
- **Métricas de desempeño**:
  - **Precisión**: LLaMA-7B ofrece una alta precisión en tareas de generación de texto, comparable a modelos de mayor tamaño, especialmente cuando se aplica fine-tuning adaptado a la tarea.
  - **Velocidad de inferencia**: Gracias a su menor número de parámetros en comparación con variantes como LLaMA-13B o LLaMA-30B, LLaMA-7B ofrece tiempos de inferencia reducidos.
  - **Consumo de memoria**: LLaMA-7B es más eficiente en términos de memoria en comparación con modelos más grandes (como LLaMA-13B o LLaMA-30B;), lo que lo hace adecuado para entornos con recursos limitados.
- **Ventajas y limitaciones**:
  - **Ventajas**:
    - **Capacidad de generación de texto**: LLaMA-7B puede generar texto coherente y de alta calidad.
    - **Equilibrio entre rendimiento y recursos**: gracias a sus 7 mil millones de parámetros, el modelo ofrece un rendimiento competitivo sin requerir infraestructuras de alto nivel. Puede ejecutarse con eficiencia en sistemas equipados con GPUs de gama media.

- **Limitaciones**:
  - **Requerimientos moderados de hardware**: Si bien es más ligero que variantes como LLaMA-13B o LLaMA-30B, LLaMA-7B sigue requiriendo hardware especializado para tareas de entrenamiento o ajuste fino. No es ideal para entornos sin GPU o con recursos extremadamente limitados.

4.2.3. Selección de técnicas de fine-tuning

Esta investigación centra su análisis experimental en cinco técnicas de ajuste fino eficiente (Parameter-Efficient Fine-Tuning o PEFT), ampliamente documentadas en la literatura científica especializada.

- **LoRA**: Low-Rank Adaptation (Hu et al., 2021)
- **QLoRA**: Quantized Low-Rank Adaptation (Dettmers et al., 2023)
- **BitFit**: Bias Fine-Tuning (Ben Zaken et al., 2021)
- **PTv2**: Prefix Tuning v2 (Li & Liang, 2021)
- ***(IA)³***: Infused Adapter by Inhibiting and Amplifying Inner Activations (X. Liu et al., 2021a)

La elección de estas metodologías se ha llevado a cabo aplicando un conjunto de consideraciones técnicas, científicas y aplicadas, que deberían permitir asegurar una cobertura metodológica completa, obtener una visión realista de la aplicabilidad de los modelos LLM preentrenados y brindar un ajuste específico al problema de la generación de consultas SQL a partir de lenguaje natural (NL-to-SQL) dentro de entornos con limitaciones computacionales.

4.2.3.1. Relevancia y actualidad científica

Todas las técnicas elegidas han sido objeto de publicaciones en conferencias de gran impacto como ICLR, ACL, NeurIPS o EMNLP, convirtiéndose en referentes dentro del área del fine-tuning eficiente.

LoRA se ha integrado ya en diferentes repositorios y frameworks de entrenamiento como Hugging Face PEFT y se convirtió, de hecho, en el método de facto para proyectos como Alpaca o Vicuna.

QLoRA se ha vuelto fundamental al facilitar el ajuste de LLMs de más de 65B parámetros en GPUs de consumo, gracias a una combinación entre cuantización cuasi-perdida y adaptadores LoRA.

Por su parte, BitFit se halla citado extensamente como baseline minimalista, contribuyendo con su perspectiva de contraste para minimizar el ajuste de modelos sin una pérdida sustancial de desempeño.

PTv2 e $(IA)^3$ constituyen líneas metodológicas que presentan elegancia conceptual y eficiencia empírica, manteniendo gran eficiencia a la vez que evitan la necesidad de grandes cambios estructurales.

4.2.3.2. Diversidad metodológica

La selección de técnicas cubre intencionadamente una diversidad arquitectónica significativa, que permite comparar no solo resultados cuantitativos sino también enfoques conceptuales distintos.

LoRA introduce adaptadores de baja dimensión dentro de las capas de atención del modelo original, reutilizando los pesos congelados del modelo base y añadiendo solo matrices entrenables de **rango reducido**.

QLoRA, al combinar cuantización de pesos a 4 bits con adaptadores LoRA, introduce un enfoque dual que optimiza tanto la carga de memoria como los parámetros actualizables.

En contraposición, BitFit se limita a ajustar únicamente los parámetros de sesgo (**bias**) de cada capa del modelo, manteniendo congelados los pesos principales, lo que representa el caso más extremo de fine-tuning minimalista.

PTv2 actúa mediante la concatenación de vectores aprendibles (**prefixes**) a las entradas *key* y *value* de las capas de atención, sin modificar ningún parámetro interno del modelo base.

Finalmente, $(IA)^3$ propone introducir tres escaladores entrenables (**input, attention y feedforward**) para cada capa transformadora, lo que permite modificar dinámicamente la importancia relativa de cada componente sin alterar su estructura.

4.2.3.3. Compatibilidad con modelos pre-entrenados

Otro criterio de selección ha sido la compatibilidad directa con modelos LLM disponibles públicamente. Todos los métodos analizados son compatibles con arquitecturas ampliamente usadas como BERT, GPT-2, Text-to-Text Transfer Transformer (T5), o LLaMA, y pueden aplicarse sin necesidad de reentrenar los modelos desde cero. Esta es una condición necesaria en aplicaciones reales donde el acceso a modelos de código abierto es el punto de partida y el entrenamiento completo no es viable ni económica ni computacionalmente. Además, métodos como LoRA, QLoRA o PTv2 cuentan con implementaciones estables y bien mantenidas en frameworks como Hugging Face Transformers y PEFT, lo que permite su reproducibilidad experimental, portabilidad y trazabilidad del código utilizado.

4.2.3.4. Eficiencia computacional

Desde una perspectiva de coste, se ha priorizado la inclusión de técnicas que reducen drásticamente el número de parámetros entrenables en comparación con el fine-tuning completo (full fine-tuning). Por ejemplo, LoRA y QLoRA consiguen reducir los parámetros entrenables a menos del 1–2% del total, mientras que BitFit lo hace hasta un 0.1%, y tanto $(IA)^3$ como PTv2 se sitúan típicamente en torno al 0.5%. Esta eficiencia se traduce no solo en menor uso de memoria VRAM, sino también en menores tiempos de entrenamiento y posibilidad de operar sobre GPUs de gama media o incluso portátiles con soporte básico de Compute Unified Device Architecture (CUDA). En particular, QLoRA demuestra que modelos de gran tamaño como LLaMA-65B se pueden ajustar con precisión competitiva usando menos de 24GB de memoria, gracias a la incorporación de métodos de cuantización activa y adaptativa.

4.2.3.5. Adecuación al caso de uso NL-to-SQL

Finalmente, la elección de estas técnicas se justifica por su adecuación específica a la tarea de generación Natural Language to SQL (NL-to-SQL), la cual requiere que el modelo aprenda

alineamientos precisos entre expresiones en lenguaje natural y estructuras semánticas complejas, como cláusulas SQL. Dado que se parte de modelos preentrenados no optimizados para este tipo de tareas estructuradas, el fine-tuning eficiente debe ser capaz de adaptar el modelo sin sobrescribir sus capacidades lingüísticas generales. Técnicas como LoRA y PTv2 han demostrado conseguir un gran desempeño en tareas que requieren precisión sintáctica, mientras que $(IA)^3$ permite un ajuste más granular de componentes individuales sin interferencia catastrófica, es decir, permitiendo el ajuste del modelo sobre nuevas tareas sin degradar el conocimiento adquirido previamente y, por tanto, sin afectar negativamente al rendimiento en tareas anteriores. QLoRA, por su parte, permite el tratamiento de modelos de mayor capacidad, lo cual es fundamental para mejorar la comprensión contextual de preguntas largas o ambigas. BitFit, si bien está más limitado en potencia expresiva, funciona como un baseline de eficiencia extrema y bajo coste, que permite calibrar la necesidad real de ajustar componentes internos en función del rendimiento observado.

En conjunto, las cinco técnicas seleccionadas ofrecen una base comparativa sólida, heterogénea y relevante, que permite no solo medir el rendimiento bajo distintos niveles de ajuste, sino también analizar la relación entre coste computacional, arquitectura modificada y precisión alcanzada en una tarea compleja y realista como NL-to-SQL. La selección ha sido, por tanto, guiada tanto por criterios científicos como por su aplicabilidad práctica en contextos donde la eficiencia no es una opción, sino una necesidad.

Tabla 4. *Resumen comparativo de técnicas PEFT*

Técnica	Tipo de adaptación	Parámetros entrenables (%)	Modifica pesos base	Ventaja principal
LoRA	Adaptadores de baja dimensión en capas de atención	~1–2%	No	Alto rendimiento con bajo coste computacional
QLoRA	Cuantización + adaptadores LoRA	~1–2%	No	Ajuste de LLMs grandes en GPUs de consumo
BitFit	Ajuste solo de parámetros de sesgo	~0.1%	No	Extrema eficiencia y simplicidad
Prefix Tuning v2	Concatenación de vectores prefix en atención	~0.5%	No	No modifica pesos y es altamente reutilizable

$(IA)^3$	Escaladores entrenables input, attention, feedforward	~0.5%	No	Control granular sin alteración estructural

Fuente: Elaboración propia.

4.2.4. Evaluación de los datos de entrada

El conjunto de datos WikiSQL (Zhong et al., 2017a) proporciona un recurso valioso para la investigación y evaluación de modelos NL-to-SQL. Este tipo de corpus representa un hito importante en la evolución de sistemas que permiten traducir preguntas en lenguaje natural a consultas estructuradas (Quamar et al., 2022), reduciendo la barrera de acceso a bases de datos por parte de usuarios no técnicos.

WikiSQL consiste en una colección de 80.654 pares de preguntas en lenguaje natural y consultas SQL asociadas, vinculadas a un total de 24.241 tablas extraídas de la Wikipedia en inglés. El dataset está dividido en tres subconjuntos: entrenamiento (56.355 instancias), validación (8.421 instancias) y prueba (15.878 instancias). Cada una de sus entradas contiene una pregunta, la consulta SQL objetivo, y una tabla a la que hace referencia dicha consulta.

Tabla 5. *Ejemplos de pares NL-to-SQL*

Pregunta (NL)	Consulta SQL generada
What is terrence ross' nationality	SELECT Nationality FROM Table WHERE Player = Terrence Ross
How many countries have more than 10 gold medals?	SELECT COUNT(*) FROM Table WHERE Gold > 10
What is the name of the player who scored the most points?	SELECT Name FROM Table WHERE Points = (SELECT MAX(Points) FROM Table)

Fuente: Kaggle.

Durante la caracterización del corpus de entrenamiento proporcionado por WikiSQL, se realizó un análisis de sus dimensiones léxicas, estructurales y estadísticas, con el objetivo de

asegurar la idoneidad del conjunto de entrenamiento y la detección de posibles sesgos o anomalías.

La estructura de las tablas en WikiSQL es altamente heterogénea. Cada tabla incluye una lista de columnas con sus respectivos nombres y una serie de registros en formato tabular. No obstante, el dataset no proporciona información explícita sobre los tipos de datos de cada columna, lo cual requiere una inferencia implícita a partir de su contenido. Esto representa un reto adicional en la fase de preparación de los datos, al requerir la implementación de mecanismos automáticos para la detección de tipos y así garantizar una representación coherente de los datos durante el entrenamiento de los modelos. El dataset, incluye metadatos asociados a cada tabla y pregunta, como identificadores, nombres de columnas normalizados, y marcas de alineación.

En la etapa de análisis exploratorio se ha contabilizado el número de columnas por tabla con una mediana de 6 columnas por tabla y valores extremos entre 5 y 44. También se ha calculado la longitud media de las preguntas en lenguaje natural, obteniendo un valor de 11,7 palabras por pregunta. No se han encontrado valores nulos de forma explícita, aunque algunas celdas contienen cadenas vacías, las cuales se han considerado como valores implícitamente faltantes. En cuanto a la detección de outliers, se ha observado la presencia de nombres de columnas excesivamente largos o con caracteres no alfabéticos, lo cual podría dificultar el pre-procesamiento del dataset.

Tabla 6. *Resumen estadístico del dataset WikiSQL: longitud de las preguntas en lenguaje natural y número de columnas por tabla*

Medida estadística	Longitud de pregunta (palabras)	Número de columnas por tabla
Media	11.7	6.3
Mediana	11	6
Máximo	58	44
Mínimo	3	5

Fuente: Elaboración propia.

4.2.4.1. Cobertura léxica

Con el objetivo de caracterizar la diversidad y estructura lingüística de las preguntas en lenguaje natural presentes en el dataset WikiSQL, se ha llevado a cabo un análisis de cobertura léxica sobre la totalidad del corpus. Para ello, se han definido cuatro categorías semántico-funcionales de interés: términos comunes, términos técnicos, construcciones interrogativas compuestas y verbos frecuentes.

Los términos comunes (por ejemplo, “who”, “what”, “how”) representan el 9,5% del total de palabras, y son fundamentales para la formulación de preguntas generales. Este porcentaje sugiere una prevalencia de estructuras interrogativas abiertas, propias de tareas de recuperación de información o consulta dirigida.

Por otra parte, los términos analíticos (como “median”, “average”, “total”) conforman tan solo el 1,0% de palabras, lo cual indica un uso muy reducido de lenguaje cuantitativo o especializado. Esta baja frecuencia es debida a la naturaleza generalista del conjunto de consultas incluidas en WikiSQL, al contrario de lo que cabría esperar de otros conjuntos de datos correspondientes a dominios técnicos.

Las construcciones compuestas interrogativas del tipo “how many”, “what is” conforman el 3,5% del conjunto, pero su identificación resulta fundamental ya que se asocian a preguntas que implican inferencias de tipo cuantitativo, recuento o comparación. Aunque su proporción, aunque no es predominante, destaca por su rol funcional al conectar lenguaje natural y estructuras SQL concretas, como el uso de cláusulas COUNT, MAX/MIN, SUM o AVG.

En cuanto a los verbos frecuentes (“is”, “are”, “does”, “have”, etc.), representan el 8,9% del total, lo cual denota un patrón interrogativo gramaticalmente simple y directo, en línea con las estructuras de WH-questions (“what”, “why”, “when”, “where”, “who”, “which”, “how”) típicas del inglés formal.

Cabe mencionar que las categorías descritas no son mutuamente excluyentes, por lo que los porcentajes individuales no suman el 100%. Por tanto, la tabla resultante se debe interpretar como un análisis de cobertura parcial, que se centra en los elementos lingüísticos estratégicos para la tarea NL-to-SQL.

Tabla 7. *Cobertura léxica en preguntas del dataset WikiSQL*

Categoría léxica	Frecuencia relativa (%)	Ejemplos
Términos comunes	9.5	who, what, how
Verbos frecuentes	8.9	is, are, does, have
Construcciones interrogativas	3.5	how many, what is
Términos técnicos	1.0	median, average, total

Fuente: Elaboración propia.

4.2.4.2. Distribución por tipo de operación SQL

Se ha cuantificado la frecuencia relativa de distintos tipos de consultas SQL presentes en el dataset. La gran mayoría incluyen cláusulas WHERE; las consultas sin condiciones son excepcionales.

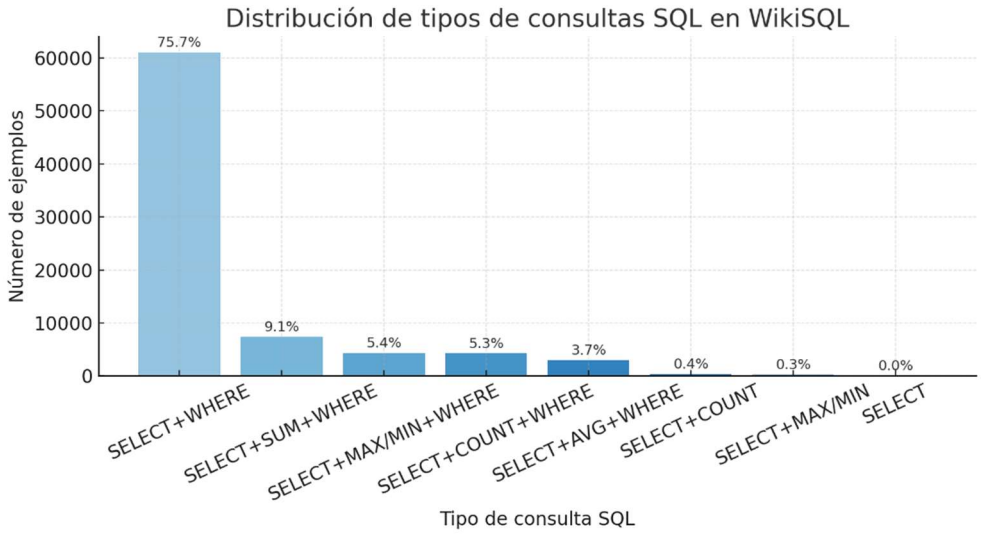
Tabla 8. *Frecuencia de tipos de consulta SQL en WikiSQL*

Tipo de consulta SQL	Frecuencia
SELECT+WHERE	61.075
SELECT+SUM+WHERE	7.347
SELECT+MAX/MIN+WHERE	4.377
SELECT+COUNT+WHERE	4.287
SELECT+AVG+WHERE	2.950
SELECT+COUNT	345
SELECT+MAX/MIN	264
SELECT	8
SELECT+SUM	1

Fuente: Elaboración propia.

La siguiente figura muestra una predominancia de las consultas simples con cláusula **SELECT + WHERE**, aunque una proporción significativa incorpora operadores de agregación (**SUM**, **MAX**/**MIN**, **COUNT**, **AVG**) y subconsultas.

Figura 16. *Distribución de tipos de consultas SQL en WikiSQL*

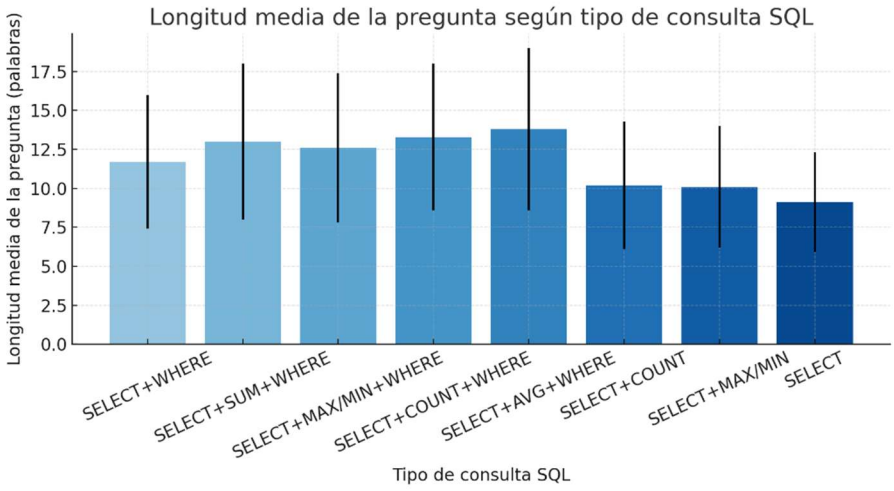


Fuente: Elaboración propia.

La siguiente figura muesta la correlación entre la longitud media de las preguntas en lenguaje natural y el tipo de consulta SQL generada. Como puede observarse:

- Las consultas simples tipo SELECT sin filtro WHERE suelen estar ligadas a preguntas más breves.
- Las consultas más complejas, como las que incluyen subqueries o aquellas que implican agregaciones (COUNT, MAX/MIN) y filtros WHERE, tienden a asociarse con preguntas más largas.

Figura 17. Distribución de tipos de consultas SQL en WikiSQL



Fuente: Elaboración propia.

4.2.4.3. Tamaño y balance de los subconjuntos

Los subconjuntos de entrenamiento, validación y prueba presentan tamaños proporcionales y no se han identificado duplicidades exactas entre ellos. No obstante, se ha detectado cierto solapamiento estructural entre tablas similares presentes en distintos subconjuntos, lo que podría introducir sesgo de evaluación si no se controla.

Tabla 9. *Distribución de instancias por partición en WikiSQL*

Partición	Número de instancias
Entrenamiento	56.355
Validación	8.421
Prueba	15.878

Fuente: Elaboración propia.

4.2.4.4. Diversidad estructural de las tablas

Para evaluar la heterogeneidad de los esquemas tabulares desde una perspectiva semántica, se llevó a cabo una clasificación de los encabezados de columna en cuatro categorías:

- Nombres estándar
- Numéricos puros
- Abreviaturas
- Nombres no significativos

La siguiente tabla ilustra la distribución de los diferentes tipos de encabezados de columna presentes en el dataset WikiSQL. Esta clasificación se ha realizado considerando exclusivamente las propiedades lingüísticas y estructurales de los encabezados, sin analizar el contenido de las celdas. **El propósito es evaluar hasta qué punto las etiquetas de columna son informativas y útiles para la tarea de mapeo semántico entre lenguaje natural y estructuras tabulares**.

El análisis revela que un 79,8% de los encabezados pueden clasificarse como nombres estándar, es decir, etiquetas con significado explícito, bien formadas y representativas de atributos semánticos relevantes (por ejemplo: “Quarterback”, “Proxy”). Esta proporción, aunque menor a la inicialmente estimada, sigue siendo dominante y facilita la alineación

semántica entre preguntas en lenguaje natural y campos tabulares, aspecto clave para la correcta generación de consultas SQL.

En cuanto a las columnas numéricas puras, es decir, aquellas cuyo encabezado es un valor numérico sin significado semántico inherente (como “1956” o “1996”), representan el 1,4 % del total. Aunque su frecuencia es reducida, estas columnas pueden suponer un desafío para los modelos de interpretación automática cuando aparecen como claves de búsqueda o criterios de agregación.

Destaca también la presencia de abreviaturas, que alcanzan un 18,8% de los encabezados. Ejemplos como “Case”, “5w” o “N.B.” reflejan la heterogeneidad y brevedad de ciertas etiquetas, las cuales pueden introducir ambigüedad en el proceso de mapeo semántico, especialmente en aquellos modelos que no han sido específicamente entrenados para manejar abreviaciones técnicas o contextuales.

Por último, la categoría de nombres no significativos (incluyendo etiquetas genéricas como “Data”) presenta una incidencia prácticamente nula. Este hecho sugiere una curación manual adecuada de los esquemas tabulares, lo cual contribuye positivamente a la calidad general del dataset y a la fiabilidad de las tareas de generación NL-to-SQL.

Tabla 10. *Distribución y ejemplos de tipos de encabezados en columnas en WikiSQL*

Tipo de columna	Proporción (%)	Ejemplos
Nombres estándar	79.8	Quarterback, Proxy, Instant messaging
Numéricas puras	1.4	1956, 1969, 1996, 1984
Abreviaturas	18.8	5w, N.B., ISBN
Nombres no significativos	~0.0	Data

Fuente: Elaboración propia.

En conjunto, **esta distribución respalda la idoneidad del dataset WikiSQL como base para entrenar modelos de NL-to-SQL en escenarios donde se espera una correspondencia semántica robusta entre lenguaje natural y estructuras de tabla**.

4.2.5. Análisis de calidad de los datos

La calidad de los datos en el dataset de WikiSQL ha sido evaluada en relación con cuatro ejes fundamentales:

- Valores atípicos
- Contenido faltante
- Sesgos lingüísticos
- Alineación semántica entre preguntas y consultas SQL

Estos aspectos son determinantes para garantizar la validez de los experimentos realizados y la generalización de los modelos entrenados.

4.2.5.1. Valores atípicos y ruido textual

El análisis de los encabezados de columna ha revelado una elevada calidad estructural del dataset WikiSQL. En concreto, un 98,6% de las columnas presentan nombres estándar, informativos y semánticamente explícitos. No se han identificado casos significativos de encabezados anómalos, como cadenas numéricas, símbolos no alfabéticos o etiquetas vacías. No obstante, se mantendrán mecanismos de detección de valores atípicos como parte del pipeline de preprocesamiento, con el fin de detectar casos puntuales que pudieran afectar a tareas de alineamiento semántico.

4.2.5.2. Valores nulos y vacíos

Tras un análisis completo del contenido de las celdas del dataset WikiSQL, no se detectó la presencia explícita de valores nulos, ni de cadenas vacías o caracteres sustitutivos (como guiones “-”) que puedan interpretarse como valores faltantes implícitos. Este resultado sugiere un alto grado de limpieza y curación de los datos en la construcción original del corpus.

No obstante, se implementarán mecanismos de validación y control de calidad en el pipeline de preprocesamiento, que permitan detectar y gestionar automáticamente cualquier anomalía en caso de extender o adaptar el dataset con nuevas fuentes. Esta estrategia debe

garantizar la consistencia semántica del conjunto de datos, incluso en escenarios donde puedan aparecer registros incompletos o no alineados en futuras ampliaciones.

4.2.5.3. Sesgos lingüísticos

Se ha observado un sesgo en la distribución sintáctica de las preguntas en lenguaje natural. Predominan las construcciones interrogativas cerradas, como "how many" o "what is", mientras que estructuras más abiertas o comparativas como por ejemplo, "which has more" o "who ranks highest", están infrarrepresentadas. Este desequilibrio podría limitar la capacidad del modelo para generalizar a tipos de consulta más expresivas, por lo que se deberán evaluar estrategias de muestreo equilibrado para futuras extensiones del estudio.

Este patrón ha sido corroborado mediante el análisis automático sobre el total de 80.654 preguntas del dataset, en el que se constató que más del 41 % contenían estructuras cerradas, mientras que menos del 0,05 % incluían construcciones abiertas o comparativas.

4.2.5.4. Consistencia pregunta-SQL

Aunque el alcance de este trabajo no contempla una verificación exhaustiva de los datos, el diseño original del dataset sí incluyó una validación humana en dos fases mediante Amazon Mechanical Turk (Zhong et al., 2017b), lo que sugiere una calidad razonablemente alta. Los análisis automáticos y el historial de uso académico de WikiSQL sugieren también una tasa de inconsistencias baja en la correspondencia semántica entre preguntas y consultas SQL. No se han identificado anomalías estructurales graves ni patrones recurrentes de desalineación durante la fase de preparación. Aun así, se analizarán estrategias para detectar y excluir ejemplos dudosos en casos donde se evidencie ambigüedad semántica o errores de anotación.

Tabla 11. *Resumen de problemas de calidad y medidas correctoras aplicadas*

Problema detectado	Impacto potencial	Solución aplicada
Predominio de estructuras tipo "how many"	Pobre generalización en preguntas complejas	Análisis de distribución y muestreo equilibrado
Celdas vacías o con guiones	Sesgo semántico en la tabla y consultas incompletas	Eliminación del ejemplo
Errores de alineación entre pregunta y SQL	Evaluación incorrecta del modelo	Exclusión del subconjunto de evaluación
Encabezados no informativos (casos aislados)	Ambigüedad en el emparejamiento semántico	Estrategias de normalización lingüística

Fuente: Elaboración propia.

4.3. Preparación de los datos (Data Preparation)

4.3.1. Selección de las Métricas de Evaluación

Una evaluación rigurosa de las técnicas de ajuste fino eficiente PEFT aplicadas a modelos LLM requiere la definición de un conjunto de métricas que aborden de forma diferenciada los objetivos fundamentales del estudio. En secciones anteriores, se han establecido dos objetivos evaluativos complementarios:

- Analizar el rendimiento de las técnicas PEFT en un entorno con recursos computacionales limitados.
- Determinar su capacidad para adaptar modelos preentrenados a la tarea específica de NL-to-SQL, utilizando el conjunto de datos WikiSQL como banco de pruebas.

Para cada uno de estos objetivos se han seleccionado métricas específicas, justificadas por su relevancia empírica.

4.3.1.1. Evaluación para el Objetivo 1: Rendimiento con recursos limitados

El primer objetivo busca caracterizar la eficiencia computacional de cada técnica PEFT en contextos donde el acceso a infraestructuras de alto rendimiento es limitado. Para ello, se han seleccionado las siguientes métricas:

- **Uso de memoria RAM (en entrenamiento e inferencia)***:* Esta métrica cuantifica el consumo de memoria (en GB) durante los procesos de entrenamiento e inferencia. Se ha obtenido mediante la monitorización continua del sistema, usando herramientas como `nvidia-smi`, `psutil` y `torch.cuda.memory_allocated()`. Su relevancia radica en que técnicas como **QLoRA** o **BitFit** prometen operar con menos del 25% del uso de memoria respecto a un ajuste completo (*full fine-tuning*). Un valor bajo indica mayor adecuación al objetivo de eficiencia.
- ***Uso de CPU/GPU****:* Se ha cuantificado la carga promedio sobre cada unidad de procesamiento durante el entrenamiento, expresada como porcentaje del uso máximo (% CPU y % GPU). La recolección de estos datos se ha realizado con `gpustat`, `nvidia-`

`smi` y `top`. Técnicas que distribuyen eficientemente la carga o que permiten prescindir del uso de GPU obtienen mejor puntuación bajo este criterio.

- ***Tiempo de entrenamiento**:* Se mide como el tiempo total transcurrido (en minutos y segundos) desde el inicio hasta la finalización de cada sesión de ajuste fino. Esta métrica es crítica para evaluar la viabilidad temporal de cada técnica, especialmente en escenarios donde se desea iterar rápidamente sobre el modelo. Un menor tiempo de entrenamiento sugiere una técnica más escalable y eficiente en término de recursos.

Los resultados obtenidos a partir de estas métricas permiten establecer una comparación objetiva entre técnicas PEFT según su coste computacional, reflejando de forma directa su aplicabilidad en contextos con infraestructuras limitadas.

4.3.1.2. Evaluación para el Objetivo 2: Precisión en la tarea NL-to-SQL

El segundo objetivo se centra en determinar la capacidad de generalización de cada técnica PEFT para adaptarse a la tarea de traducción de lenguaje natural a SQL. Para este fin, se han seleccionado métricas que evalúan tanto la corrección sintáctica como semántica de las consultas generadas, así como el desempeño computacional durante la inferencia:

- **Exactitud literal (Exact Match)***:* Esta métrica indica el porcentaje de consultas generadas que coinciden exactamente con la cadena SQL de referencia, sin considerar equivalencias semánticas. Se calcula mediante una comparación directa entre la salida del modelo y la consulta objetivo. Aunque estricta, esta métrica permite establecer una línea base clara y objetiva.

$$\text{Exact Match} = \frac{\text{Número de coincidencias exactas}}{\text{Total de ejemplos evaluados}} * 100$$

- **Tiempo medio de inferencia por muestra***:* Además de su utilidad para el Objetivo 1, esta métrica se considera aquí como indicador del coste operativo en producción. Una técnica que incrementa significativamente la latencia por predicción, aunque aumente la precisión, podría ser inviable en sistemas interactivos.

En conjunto, estas métricas ofrecen una visión integral del rendimiento de cada técnica, abordando tanto la dimensión *funcional* (precisión) como la *operativa* (eficiencia). Su

combinación permite comparar no solo qué técnica genera las mejores respuestas, sino también cuál lo hace de forma más económica y sostenible desde el punto de vista computacional.

4.3.2. Preparación del entorno de prueba

Para llevar a cabo la evaluación de las técnicas PEFT seleccionadas bajo condiciones controladas y reproducibles, se ha preparado un entorno experimental que permite la ejecución eficiente del fine-tuning de modelos LLM, así como la recolección sistemática de métricas de rendimiento. Este entorno consta de dos componentes principales: la infraestructura de hardware en la nube necesaria para entrenar los modelos y la implementación de un sistema automatizado de experimentación.

4.3.2.1. Infraestructura de ejecució

Para poder disponer de un entorno con los recursos computacionales limitados pero realistas, se ha contratado un servidor dedicado **GEX44 con GPU**. Este servidor proporciona una plataforma robusta para llevar a cabo tareas de entrenamiento e inferencia de modelos LLM de manera controlada.

Las características técnicas de la infraestructura utilizada son las siguientes:

- **Proveedor**: Hetzner *(https://www.hetzner.com/)*
- **Servidor dedicado**: GEX44 GPU (generación Ada)
- **Procesador**: Intel® Core™ i5-13500 (13ª generación Raptor Lake)
    - 6 *Performance Core* y 8 *Efficiency Cores*.
    - Hyper-Threading y virtualización Intel-VT
- **Memoria RAM**: 64 GB DDR4
- **GPU**: NVIDIA RTX™ 4000 SFF Ada Generation
    - 20 GB de memoria GDDR6 ECC
- **Almacenamiento**: 2 × 1.92 TB Gen3 NVMe SSD (Datacenter Edition)
- **Sistema operativo**: Ubuntu 64 bits.
- **Acceso y administración**:
    - IP dedicada IPv4 y subred IPv6 (/64)

- Protección DDoS, firewall configurable y estadísticas de tráfico
- Alta disponibilidad de red (99.9 %) con tecnología de enrutamiento Juniper

Esta infraestructura ofrece un equilibrio entre capacidad de cómputo y consumo eficiente de energía (33W en reposo y hasta 187W en carga máxima), permitiendo la ejecución de modelos LLM de tamaño medio con un consumo optimizado.

4.3.2.2. Implementación del entorno experimental

Con el objetivo de garantizar la **portabilidad**, **reproducibilidad** y **escalabilidad** de los experimentos, se ha optado por una arquitectura basada en **contenedores Docker**. Esta elección permite encapsular todas las dependencias necesarias para el entrenamiento y evaluación de modelos LLM con técnicas PEFT, asegurando coherencia en los resultados, independencia del entorno local y facilidad de despliegue en servidores remotos con GPU.

Ventajas de la solución basada en Docker:

- Reproducibilidad garantizada, ya que los experimentos son replicables en cualquier máquina compatible con Docker y GPU, sin necesidad de configuración adicional.
- Aislamiento del entorno, para evitar conflictos de dependencias y asegurar que cada experimento se ejecute en condiciones idénticas.
- Flexibilidad y extensibilidad, ya que se pueden añadir fácilmente nuevas técnicas PEFT o modelos al entorno actual con mínimas modificaciones.
- Monitoreo de recursos incorporado, para recopilar métricas detalladas durante cada ejecución, fundamentales para los objetivos del estudio.

4.3.2.3. Arquitectura basada en Docker

Se ha construido una imagen Docker personalizada diseñada específicamente para soportar el entrenamiento eficiente de modelos de lenguaje con fine-tuning parcial (PEFT) en entornos con recursos computacionales limitados. Esta imagen contiene todos los componentes requeridos para:

- Ejecutar entrenamientos acelerados por GPU mediante CUDA 12.6.
- Implementar técnicas de fine-tuning eficiente como *LoRA, QLoRA, Prefix Tuning v2 e (IA)³*.

- Evaluar el rendimiento de los modelos en tareas de traducción de lenguaje natural a SQL.

4.3.2.4. Componentes principales de la imagen Docker

La imagen incluye un conjunto específico de librerías y herramientas necesarias para realizar los experimentos:

- **PyTorch 2.7 con soporte CUDA 12.6**: núcleo del entrenamiento de modelos, aprovechando las capacidades de cómputo de la GPU NVIDIA RTX 4000 Ada.
- **Transformers (Hugging Face)**: para cargar modelos LLM preentrenados y gestionar tokenizadores, entrenamientos y generación de texto.
- **Datasets y Evaluate (Hugging Face)**: para manejar conjuntos de datos estandarizados y calcular métricas de evaluación.
- **Librería PEFT (Hugging Face)**: repositorio oficial de Hugging Face con implementación de múltiples técnicas PEFT, incluyendo *LoRA*, *QLoRA*, *Prefix Tuning v2* e *(IA)³*.
- **BitsAndBytes**: permite cuantización de modelos (por ejemplo, 4-bit) para ahorrar memoria y facilitar la aplicación de técnicas como *QLoRA*.
- **Accelerate**: para gestionar el entrenamiento distribuido y optimizado en GPU o CPU de forma sencilla.
- **SentencePiece y Protobuf**: utilizados para el soporte en tokenización y serialización eficiente.
- **psutil y pynvml**: para monitorear en tiempo real el uso de CPU, memoria RAM y memoria GPU durante el entrenamiento e inferencia.

4.3.2.5. Funcionalidades de la imagen Docker

El sistema permite ejecutar entrenamientos controlados con técnicas PEFT sobre modelos LLM, integrando selección de técnica y modelo, carga de datos, monitoreo de recursos, evaluación automatizada y registro estructurado de resultados, todo ello mediante una arquitectura flexible y reproducible orientada a tareas de traducción de lenguaje natural a SQL.

- **Selección de técnica PEFT**: El sistema permite elegir entre *LoRA*, *QLoRA*, *Prefix Tuning v2* o *(IA)³* mediante argumentos de línea de comandos o archivos de configuración.
- **Selección de modelo LLM**: Se puede seleccionar el modelo base (GPT-2, LLaMA-7B) usando las interfaces de Hugging Face.
- **Carga de datasets**: Incluye soporte para la carga y preprocesamiento de datasets de traducción de lenguaje natural a SQL, como *Salesforce/wikisql*.
- **Monitoreo de recursos**: El sistema integra herramientas como `psutil` y `nvidia-smi` para registrar el uso de CPU, GPU y memoria RAM durante el entrenamiento y la inferencia.
- **Evaluación automatizada**: Calcula métricas de *Exactitud literal* y *Tiempo de inferencia* para evaluar el rendimiento del modelo fine-tuneado en tareas NL → SQL.
- **Almacenamiento de resultados**: Los logs y métricas de cada experimento se almacenan automáticamente en archivos estructurados (JSON), permitiendo análisis posteriores.

4.3.2.6. Estructura del contenedor

La estructura del contenedor está diseñada para facilitar la automatización y configuración de los experimentos de fine-tuning con técnicas PEFT. En el directorio src se encuentra el código Python que orquesta todo el proceso de entrenamiento, desde la carga del modelo y los datos hasta la aplicación de la técnica seleccionada y la evaluación del rendimiento. Por otro lado, el directorio `configs` contiene los archivos JSON que definen los hiperparámetros del entrenamiento y los parámetros específicos de cada técnica PEFT utilizada. A continuación, se describen algunos de los componentes principales incluidos en esta estructura.

4.3.2.7. `Src/train.py`

Tiene como responsabilidad orquestar el entrenamiento de un modelo LLM con una técnica PEFT seleccionada, aplicando fine-tuning sobre el dataset WikiSQL, en el contexto de una tarea de traducción de lenguaje natural a SQL. A continuación, se detalla su funcionalidad por etapas:

- **Selección y configuración de la técnica PEFT**: El script inicia determinando la técnica PEFT que se utilizará en el entrenamiento, en función del argumento `--peft`

proporcionado al ejecutarlo. Puede seleccionarse entre varias opciones: *LoRA*, *QLoRA* (que incluye cuantización en 4 bits mediante `BitsAndBytes`), *Prefix Tuning* o *(IA)³*. Cada técnica está configurada a partir de un archivo JSON específico ubicado en el directorio `configs/`, donde se definen los parámetros particulares que se adaptan tanto al modelo base como a la tarea de ajuste fino, en este caso centrada en *language modeling causal* (`CAUSAL_LM`).

- **Carga del modelo base**: Una vez determinada la técnica de fine-tuning, se procede a la carga del modelo base desde Hugging Face, pudiendo ser modelos como GPT-2 o LLaMA. Este modelo se inicializa con configuraciones avanzadas, incluyendo la cuantización en QLoRA o la selección específica de módulos a modificar en técnicas como LoRA o IA³. De forma complementaria, se carga también el *tokenizer* correspondiente, necesario para el preprocesamiento del texto de entrada según el modelo elegido.
- **Preparación del dataset**: A continuación, se prepara el conjunto de datos que se utilizará para el entrenamiento y la evaluación. En este caso, se utiliza *WikiSQL*, un dataset diseñado para tareas de traducción de lenguaje natural a consultas SQL. Este dataset se preprocesa y adapta mediante funciones específicas que tokenizan las entradas de acuerdo con el modelo seleccionado.
- **Configuración del entrenamiento**: Posteriormente, se configuran los parámetros de entrenamiento, los cuales se cargan desde un archivo JSON externo. Estos parámetros se utilizan para inicializar un objeto `Trainer` de la librería Hugging Face, al que se le añade un *callback* personalizado (`StepEvalAndEarlyStopCallback`) que permite realizar evaluaciones periódicas durante el entrenamiento y aplicar una lógica de parada anticipada si se detecta estancamiento.
- **Ejecución del entrenamiento**: Con todos los elementos listos, se da comienzo al entrenamiento. Antes de iniciar, se activa un sistema de monitorización de recursos mediante `start_benchmark_metrics()`, lo que permite registrar el uso de CPU, GPU y memoria durante la ejecución. Luego, se realiza el fine-tuning del modelo con la técnica PEFT seleccionada, y al finalizar, se detiene la monitorización para guardar los resultados de consumo computacional junto con los registros del experimento.

- **Persistencia del modelo adaptado**: Finalmente, se guarda el adaptador PEFT resultante en la ruta especificada por `config.adapter_dir`, lo que permite reutilizar o evaluar el modelo ajustado en futuras fases del proyecto sin necesidad de reentrenar.

Este script automatiza el entrenamiento de un modelo LLM con técnicas PEFT sobre tareas de generación de SQL a partir de lenguaje natural, permitiendo comparar técnica por técnica bajo condiciones homogéneas, midiendo rendimiento, eficiencia y capacidad adaptativa.

4.3.2.8. Src/infer.py

`infer.py` proporciona una interfaz interactiva en línea de comandos que permite evaluar de forma práctica un modelo de lenguaje ajustado con técnicas PEFT, aplicado a la tarea de traducción de lenguaje natural a SQL. Su principal objetivo es permitir al usuario introducir preguntas en lenguaje natural, junto con un conjunto de nombres de columnas, y obtener como resultado una consulta SQL generada por el modelo fine-tuneado. A continuación, se detalla su funcionalidad por etapas:

- **Carga del modelo y del adaptador PEFT**: El proceso comienza con la carga de los argumentos de entrada, que indican qué modelo utilizar y qué técnica PEFT ha sido aplicada. A continuación, se cargan las variables de entorno necesarias, como el token de acceso a Hugging Face, y se selecciona la configuración adecuada para el modelo y la técnica especificada. También se inicializa el tokenizer correspondiente al modelo, necesario para convertir el texto de entrada en tensores que puedan ser interpretados por el modelo.
- **Bucle interactivo de inferencia**: Una vez listo el modelo, el sistema entra en un bucle interactivo. El usuario introduce una pregunta en lenguaje natural y una lista de columnas de una tabla, separadas por comas. Este prompt se tokeniza y se envía al modelo, que genera una salida en forma de texto con una posible sentencia SQL correspondiente a la consulta planteada. Para la generación, se utilizan parámetros como `temperature`, `top_k` y `top_p` que controlan la creatividad y diversidad de las respuestas.

- **Generación y visualización del resultado**: Finalmente, la salida generada se decodifica y se muestra en pantalla. Este proceso se repite hasta que el usuario decide finalizar la sesión introduciendo el comando `/quit`.

Este script cumple la función de **interfaz de evaluación manual**, permitiendo verificar de forma práctica y flexible cómo responde el modelo fine-tuneado a nuevas preguntas en lenguaje natural. Es una herramienta útil para comprobar la calidad cualitativa de la generación SQL más allá de métricas automáticas.

4.3.2.9. `src/wikisql_dataset.py`

El módulo `wikisql_dataset.py` se encarga de gestionar todo el ciclo de preparación del conjunto de datos **WikiSQL**, desde su carga hasta su adaptación final para entrenar modelos de lenguaje orientados a la generación de consultas SQL a partir de texto en lenguaje natural.

Esta funcionalidad incluye la tokenización, la construcción de prompts estructurados y el enmascaramiento de partes no supervisadas del texto, adaptando los datos a los requisitos de entrenamiento con técnicas PEFT. Su diseño modular y reutilizable permite integrar fácilmente este preprocesamiento en distintos experimentos con modelos LLM.

El detalle completo de este proceso se describe en la siguiente *sección: **"Pre-procesamiento de los datos"***.

4.3.2.10. `Configs/`

El directorio `configs` contiene los archivos de configuración utilizados para parametrizar tanto los entrenamientos como las distintas técnicas. Estos ficheros permiten ajustar dinámicamente los valores de hiperparámetros relevantes sin necesidad de modificar el código fuente ni reconstruir la imagen Docker, facilitando así la experimentación flexible y controlada.

Cada archivo define una configuración específica asociada a una técnica PEFT concreta. A continuación, se enumeran los archivos disponibles en este directorio:

- **`training_configuration.json`**: Contiene los hiperparámetros generales del proceso de entrenamiento.
- **`peft_ia3_configuration.json`**: Contiene la configuración mínima para aplicar la técnica *(IA)³*.
- **`peft_lora_configuration.json`**: Define los parámetros de *LoRA*.
- **`peft_prefix_configuration.json`**: Configura la técnica *Prefix Tuning*.
- **`peft_qlora_configuration.json`**: Incluye parámetros relacionados con la cuantización en 4 bits para *QLoRA*.

Este enfoque basado en archivos de configuración externos mejora la trazabilidad y reproducibilidad de los experimentos, permitiendo cambios rápidos en los parámetros sin alterar la infraestructura subyacente ni el código de entrenamiento.

4.3.3. Pre-procesamiento de los datos

El preprocesamiento de los datos es un paso esencial en un pipeline de entrenamiento de modelos de lenguaje, especialmente en contextos de fine-tuning con modelos de gran escala como GPT-2 y LLaMA-7B. En esta sección se detalla el procedimiento seguido para transformar el dataset WikiSQL en un formato compatible con dichos modelos, que optimice tanto la eficiencia computacional como la calidad del entrenamiento.

4.3.3.1. Descripción del dataset WikiSQL

El dataset empleado, *Salesforce/wikisql*, está disponible públicamente a través de Hugging Face Datasets y contiene tres particiones estáticas y predefinidas: **train** (56.355 muestras), **validation** (8.421 muestras) y **test** (15.878 muestras). Estas particiones se mantienen inalteradas a lo largo del tiempo siempre que se use el mismo commit del repositorio oficial. Por tanto, para garantizar la reproducibilidad de los experimentos (Pineau et al., 2021) y evitar discrepancias involuntarias entre ejecuciones, se fijó explícitamente el commit del dataset en los scripts de carga. La fijación del commit se realiza especificando el parámetro revision al usar `load_dataset`, de la siguiente forma:

```
load_dataset("Salesforce/wikisql", revision="<commit_hash>")
```

4.3.3.2. Análisis estadístico de las longitudes de secuencia

Antes de definir los valores óptimos para el parámetro `max_length`, se llevó a cabo un análisis exhaustivo de la longitud en tokens de las dos secuencias principales procesadas por el modelo: el prompt (compuesto por una pregunta en lenguaje natural y los nombres de columnas de la tabla) y la consulta SQL de referencia. Este análisis se realizó mediante el script `data_analysis.py`, utilizando los tokenizadores de `gpt2` y `meta-llama/Llama-2-7b-hf` respectivamente.

Los resultados obtenidos para el split de entrenamiento y validación se resumen en la siguiente tabla tipificada:

Tabla 12. *Análisis estadístico de las longitudes de entrada*

Modelo	Split	Tipo	Máximo	P99	P95	Mediana	Media
GPT-2	train	Prompt	197	85	62	40	42.4
GPT-2	train	SQL	147	33	25	14	15.0
GPT-2	validation	Prompt	165	78	60	40	42.1
GPT-2	validation	SQL	76	32	25	14	14.9
LLaMA-7B	train	Prompt	423	116	76	45	48.2
LLaMA-7B	train	SQL	192	42	31	17	18.3
LLaMA-7B	validation	Prompt	244	109	73	45	47.7
LLaMA-7B	validation	SQL	111	41	31	17	18.2

Fuente: Elaboración propia.

A partir de este análisis se dedujeron los valores apropiados para `PROMPT_MAX_LENGTH` y `SQL_MAX_LENGTH` siguiendo las recomendaciones del estado del arte para evitar tanto

truncamiento excesivo como padding innecesario, lo cual podría afectar la eficiencia de entrenamiento y el aprovechamiento de la VRAM del sistema (Hugging Face, 2022).

4.3.3.3. Selección de longitudes óptima

Para cubrir el 99 % de los datos, se tomaron los percentiles 99 de longitud de cada segmento y se añadió un pequeño margen de seguridad antes de redondear a múltiplos de 8, como recomienda la documentación técnica de Hugging Face para maximizar el rendimiento en GPU.

- **`PROMPT_MAX_LENGTH = 96`** (GPT-2) o **128** (LLaMA-7B). Cubre ~99 % de los casos, hasta el percentil 99 + margen de seguridad, redondeado a múltiplo de 8.
- **`SQL_MAX_LENGTH = 40`** (GPT-2) o **48** (LLaMA-7B). Cubre ~99 % de las consultas SQL, hasta el percentil 99 + margen, redondeado a múltiplo de 8)

Esta configuración logra truncar solo el 1% de los ejemplos más largos, manteniendo un padding medio razonable: ~54 tokens para el prompt y ~25 para la consulta SQL. Estos valores se integraron en el script `wikisql_dataset.py` durante la tokenización de los ejemplos.

4.3.3.4. Preprocesamiento de ejemplos y tokenización

La clase Python `wikisql_dataset.py` implementa las funciones necesarias para construir prompts y etiquetas en formato texto y tokenizarlos de forma coherente con el modelo elegido. Cada ejemplo del dataset es procesado para generar dos entradas:

- **Prompt de entrada**: Texto de entrada con la pregunta `question` en lenguaje natural (inglés) más los `['table']['header']` de las columnas de la tabla. Con el siguiente formato:

```
Question: {question}
Columns: {header_1}, {header_2}, ...
SQL:
```

- **Etiqueta objetivo**: Consulta SQL en formato texto extraída de `['sql']['human_readable']`

Ambas entradas son tokenizadas con el modelo correspondiente y convertidas en tensores. El prompt se tokeniza con `padding='max_length'` y `truncation=True`, utilizando el valor de `PROMPT_MAX_LENGTH` seleccionado. Las etiquetas `human_readable_sql` se tokenizan por separado, utilizando `SQL_MAX_LENGTH`.

Asimismo, se establece `tokenizer.pad_token = tokenizer.eos_token` cuando es necesario, por ejemplo, en el modelo GPT-2 para asegurar que los tokens de padding no interfieran en la inferencia ni en el cálculo de pérdidas. En caso de utilizar LLaMA, se emplea la función `prepare_seq2seq_batch` con la opción de truncamiento activada (Raffel et al., 2020).

4.3.3.5. Consideraciones finales

El estudio de distribuciones de longitud previo al entrenamiento permite realizar una selección informada de los valores de truncamiento y padding, ajustando los recursos computacionales disponibles a las necesidades reales del dataset. Al adaptar los valores de longitud de entrada a los percentiles superiores con un margen razonable, se logra un equilibrio óptimo entre cobertura de datos y eficiencia computacional. Esta estrategia, descrita en la literatura de fine-tuning de LLMs (Raffel et al., 2020)(Lv et al., 2023) , evita tanto el truncamiento indiscriminado de ejemplos relevantes como el desperdicio de capacidad de procesamiento en padding innecesario.

Además, la estructura clara del prompt y la separación entre entrada y etiqueta permiten una preparación robusta de los datos de entrada para los modelos.

4.4. Modelado (Modeling)

Como parte de la evaluación de las distintas técnicas PEFT, se han llevado a cabo entrenamientos utilizando los modelos GPT-2 y LLAMA-7B como base, aplicando sobre ellos las técnicas seleccionadas. Los ajustes finos de los modelos se han llevado a cabo empleando los parámetros por defecto proporcionados por cada técnica, con el objetivo de establecer un baseline inicial sobre el que comparar futuras optimizaciones.

Los resultados presentados en los siguientes apartados responden a los objetivos 1 y 2 definidos en el planteamiento metodológico del trabajo. El estudio comparativo se estructura

en torno a: (1) analizar el impacto de cada técnica en términos de uso de recursos computacionales, incluyendo tiempos de entrenamiento, consumo de memoria y utilización de GPU/CPU; Para ello, durante el proceso de entrenamiento, donde se aplicaron las distintas técnicas de ajuste fino sobre los modelos LLM, se recopilaron datos sobre varios aspectos de este, entre ellos:

- **Train Time**: Tiempo total de entrenamiento.
- **Train Steps/s**: Pasos de entrenamiento por segundo.
- **GPU Mem**: Memoria de GPU utilizada.
- **GPU Util**: % de utilización de GPU.
- **CPU Util**: % de utilización de CPU.
- **Epochs**: Epochs de entrenamiento.
- **Train Loss**: Función de pérdida en datos de entrenamiento.
- **Eval Loss**: Función de pérdida en datos de evaluación.
- **Grad Norm**: Gradiente de normalización.
- **Learning Rate**: Tasa de aprendizaje.

Y (2) comparar el rendimiento de cada técnica en la tarea específica de traducción de lenguaje natural a SQL, evaluando métricas como la pérdida en entrenamiento y validación, la norma del gradiente, tasas de aprendizaje y resultados de inferencia.

En esta tarea, se pretende que el modelo transforme correctamente una pregunta en lenguaje natural en una consulta SQL válida y semánticamente correcta. Para cuantificar la precisión, los resultados de inferencia se han clasificado en las siguientes categorías:

- **Match**: Existe una coincidencia exacta entre la consulta SQL generada por el modelo y la consulta SQL original presente en el dataset. Representa el caso ideal y se indica en las gráficas con color verde.
- **Projection mismatch**: El modelo ha generado incorrectamente la cláusula de proyección (*SELECT*, es decir, las columnas seleccionadas en la consulta).
- **Filter mismatch**: El modelo ha cometido errores en la cláusula de filtrado (*WHERE*), seleccionando condiciones erróneas o irrelevantes.

- **Both projection and filter mismatch**: Existen errores en ambas partes de la consulta, tanto en la proyección como en el filtro.
- **Invalid SQL format**: La consulta generada no tiene una sintaxis válida o directamente no puede interpretarse como una consulta SQL ejecutable.

Antes de continuar con el detalle de la comparativa, debemos hacer mención a la no implementación de la técnica de PEFT conocida como BitFit, debido a su limitada integración en las bibliotecas estándar de Hugging Face, particularmente en su librería PEFT.

Aunque BitFit es una técnica reconocida por su simplicidad y eficiencia, su soporte en el ecosistema de Hugging Face es parcial y no está completamente integrado en las interfaces de alto nivel proporcionadas por PEFT. Se observa que las técnicas actualmente soportadas incluyen LoRA, QLoRA, Prefix Tuning e $(IA)^3$, entre otras, pero BitFit no se encuentra entre las opciones directamente disponibles. Aunque existen implementaciones de BitFit en repositorios independientes y algunos modelos en el Hub de Hugging Face que han sido ajustados utilizando esta técnica, la falta de una integración oficial y mantenida en las bibliotecas principales representa un desafío para su adopción en proyectos que buscan mantener compatibilidad y facilidad de uso con las herramientas estándar.

Implementar BitFit requeriría una modificación manual de los scripts de entrenamiento, incluyendo la congelación selectiva de parámetros y la gestión personalizada de los términos de sesgo, lo cual podría introducir complejidad adicional y potenciales errores en el proceso de ajuste fino. Dado que el objetivo de este trabajo es evaluar técnicas de ajuste fino eficientes en parámetros que sean fácilmente reproducibles y compatibles con entornos de recursos limitados, se ha decidido centrarse en aquellas metodologías que cuentan con soporte oficial y documentación detallada en el ecosistema de Hugging Face.

En resumen, la exclusión de BitFit de esta comparativa se debe a su limitada integración en las herramientas estándar utilizadas, priorizando técnicas que ofrecen una implementación más directa y soporte activo en la comunidad, facilitando así la reproducibilidad y aplicabilidad de los resultados obtenidos.

4.5. Evaluación (Evaluation)

En esta sección se presentan los resultados obtenidos a partir de la aplicación de diversas técnicas de ajuste fino sobre dos modelos de lenguaje representativos: GPT-2 y LLaMA-7B. Este análisis permite identificar fortalezas y limitaciones de cada enfoque, proporcionando una visión razonada y empírica sobre su idoneidad de aplicación.

4.5.1. Modelo GPT-2

El modelo GPT-2, con aproximadamente 1.5 mil millones de parámetros, representa un punto intermedio entre modelos ligeros y grandes LLMs, lo que lo convierte en una base adecuada para evaluar la eficacia de las técnicas PEFT en contextos de entrenamiento menos exigentes computacionalmente. En esta subsección se analiza el comportamiento de las técnicas PEFT aplicadas sobre GPT-2, tanto desde la perspectiva del uso de recursos como del rendimiento en la tarea de traducción de lenguaje natural a SQL.

4.5.1.1. Análisi de rendimiento con recursos limitados (Objetivo 1)

El análisis comparativo del consumo de recursos computacionales en los entrenamientos realizados sobre el modelo GPT-2 utilizando las técnicas de ajuste fino muestraron diferencias notables en términos de tiempo de entrenamiento, velocidad de ejecución en pasos por segundo y memoria GPU consumida.

Tabla 13. *Recursos computacionales y eficiencia temporal por técnica PEFT sobre GPT-2*

Técnica	Train Time	Train Steps/s	GPU Mem (MB)	GPU Util. (%)	CPU Util. (%)
LoRA	00:49:50	11.78	2013	92.61	5.0
QLoRA	00:47:54	12.26	1587	91.62	5.0
Prefix	03:05:50	3.16	2084	92.84	4.99
$(IA)^3$	03:07:55	3.13	2038	93.38	5.0

Fuente: Elaboración propia.

En primer lugar, atendiendo al tiempo de entrenamiento, las técnicas LoRA y QLoRA presentan un tiempo de entrenamiento similar y significativamente inferior al de Prefix Tuning e $(IA)^3$, con tiempos por debajo de los 50 minutos frente a las algo más de 3 horas requeridas por Prefix Tuning e $(IA)^3$. Esta diferencia se explica principalmente por la naturaleza intrínseca de cada técnica, ya que tanto LoRA como QLoRA introducen matrices de bajo rango para adaptar

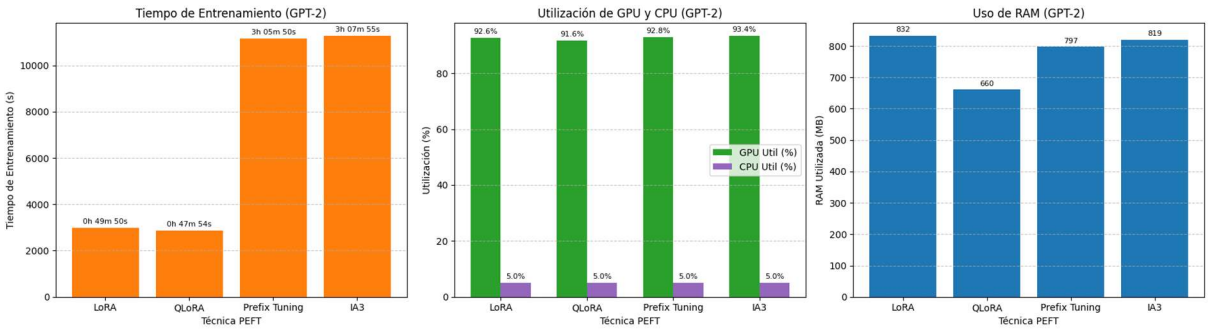
parcialmente los pesos de atención del modelo, lo que les permite ser computacionalmente más eficientes. Destaca particularmente QLoRA que, además, utiliza cuantización de pesos, reduciendo aún más la carga computacional en entrenamiento y memoria requerida. Esta cuantización también se refleja en la reducción de memoria GPU utilizada por QLoRA (1.5GB) frente al resto de técnicas, que es de 2GB aproximadamente.

Prefix Tuning e $(IA)^3$ muestran tiempos de entrenamiento notablemente más largos debido a la forma en que operan. Prefix Tuning implica aprender nuevos tokens de contexto en cada paso y, aunque mantiene los pesos originales congelados, la adición y aprendizaje de prefijos consume más tiempo computacional y ligeramente más memoria GPU. Por su parte, $(IA)^3$ ajusta individualmente parámetros escalares para módulos específicos del modelo, generando también un entrenamiento considerablemente más lento.

En términos de velocidad de ejecución de pasos por segundo, nuevamente LoRA y QLoRA destacan sobre el resto, superando en casi cuatro veces a Prefix Tuning e $(IA)^3$. Este comportamiento estaría directamente correlacionado con el menor número de parámetros efectivos modificados en las técnicas LoRA y QLoRA.

Todas las técnicas mantienen una utilización de GPU bastante alta (por encima del 90%), indicando que la GPU es aprovechada al máximo. Por otro lado, la utilización de CPU permanece baja y prácticamente constante (en torno al 5%) para todas las técnicas, lo que evidencia que el cuello de botella de estos procesos no está en la CPU, sino en la capacidad de cómputo y memoria de la GPU.

Figura 18. *Comparativa uso de recursos técnicas PEFT sobre el modelo GPT-2*



Fuente: Elaboración propia.

LoRA y en especial QLoRA destacan claramente en eficiencia temporal y consumo de memoria GPU, lo que las convierte en ideales en entornos con recursos limitados.

4.5.1.2. Análisis de precisión en la tarea NL-to-SQL (Objetivo 2)

En términos de rendimiento en la tarea de traducción de NL a SQL, LoRA obtiene el resultado más bajo para la función de pérdida (0.0739), lo que indica un rendimiento sólido en términos generales. La combinación de un loss bajo, tanto en entrenamiento como en evaluación, refleja una excelente capacidad de generalización del modelo utilizando esta técnica, minimizando considerablemente el sobreajuste.

QLoRA presenta un rendimiento muy similar a LoRA, con un valor en función de pérdida ligeramente superior. Considerando su menor consumo de memoria, esta técnica ofrece una excelente relación eficiencia-rendimiento, muy equilibrada para escenarios con recursos limitados.

Prefix Tuning muestra un valor de pérdida en validación superior al resto (0.1635). Junto al alto valor de pérdida en entrenamiento también sugiere una dificultad del modelo para ajustarse adecuadamente a los datos proporcionados, potencialmente por la complejidad del entrenamiento adicional de los prefijos. Esta técnica requeriría una búsqueda de hiperparámetros más exhaustiva para obtener mejores resultados.

$(IA)^3$ presenta unos valores de pérdida intermedios. A pesar de ser más lento, su rendimiento no supera a técnicas más rápidas como LoRA y QLoRA. Por tanto, su aplicabilidad debería estar justificada en casos específicos donde su adaptación estructural aporte ventajas concretas sobre los métodos anteriores.

Tabla 14. *Métricas de entrenamiento por técnica PEFT sobre GPT-2*

Técnica	Epoch	Train Loss	Eval Loss	Grad Norm	Learning Rate
LoRA	2.64	0.2226	0.0739	0.3643	3.68e-5
QLoRA	2.36	0.2502	0.0782	0.3550	3.82e-5
Prefix	9.99	0.5425	0.1635	0.0483	6.39e-8
$(IA)^3$	9.96	0.3971	0.1522	0.0798	2.04e-7

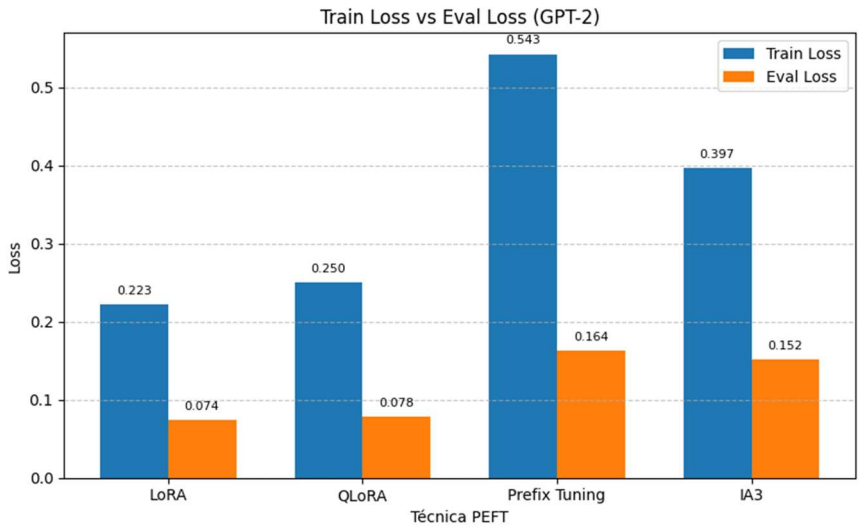
Fuente: Elaboración propia.

Respecto al gradiente de normalización, se observa un patrón claro: las técnicas con mayor adaptación estructural como LoRA y QLoRA presentan valores más altos (0.3643 y 0.3550 respectivamente). La mayor magnitud promedio en los gradientes se traduce en una

adaptación más rápida y estable a los datos de entrenamiento. Por el contrario, Prefix Tuning e $(IA)^3$ presentan gradientes menores (0.0483 y 0.0798 respectivamente) y muestran menos adaptabilidad directa, lo que ralentiza la convergencia durante el entrenamiento.

La tasa de aprendizaje elegida por defecto es considerablemente más alta en LoRA y QLoRA, lo cual se correlacionaría con el rendimiento más eficiente en términos de tiempo y calidad de aprendizaje. Prefix e $(IA)^3$, con tasas de aprendizaje mucho menores, requieren muchos más epochs para alcanzar niveles similares de convergencia.

Figura 19. *Comparativa función de pérdida técnicas PEFT sobre el modelo GPT-2*



Fuente: Elaboración propia.

4.5.1.3. Resultados de precisión en inferencia

Para este análisis de precisión en inferencia se utilizó un conjunto de evaluación con 8.421 muestras pertenecientes al conjunto de validación del dataset WikiSQL, evaluando la calidad de las consultas SQL generadas por el modelo.

Las siguientes figuras resumen la distribución de los resultados por técnica PEFT aplicada sobre GPT-2:

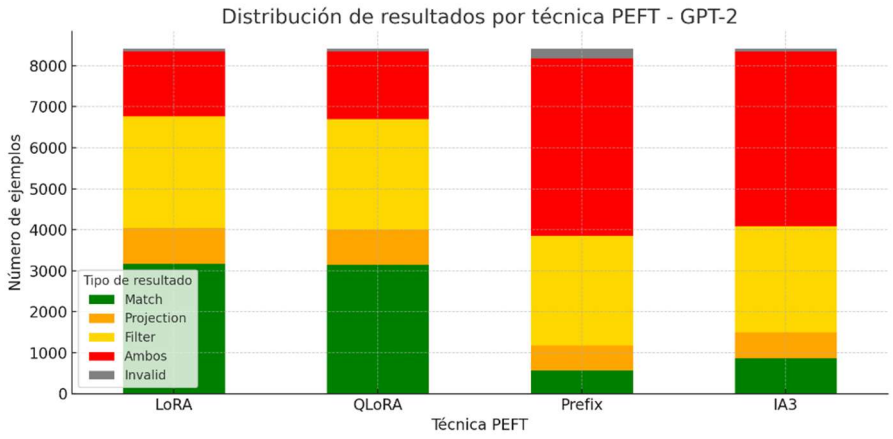
Tabla 15. *Distribución de resultados por técnica PEFT (GPT-2)*

Técnica	Match	Projection	Filter	Ambos	Invalid
LoRA	3177	871	2718	1585	70
QLoRA	3143	861	2688	1663	66

Prefix	567	605	2676	4329	244
(IA)³	862	631	2590	4263	75

Fuente: Elaboración propia.

Figura 20. *Distribución de resultados por técnica PEFT (GPT-2)*



Fuente: Elaboración propia.

LoRA y QLoRA destacan como las técnicas más eficaces, con unas tasas de acierto exacto cercanas al 38%. En ambos casos, los errores más frecuentes se concentran en la cláusula *WHERE*, lo que indica mayor dificultad del modelo para aprender condiciones de filtrado precisas. La proporción de errores en la proyección está en unos valores muy por debajo.

Prefix tuning e (IA)³ muestran un rendimiento mucho más bajo en términos de aciertos exactos, con apenas un 6.7% y 10.2% respectivamente. En ambos casos, domina la categoría de errores combinados, es decir, errores tanto en la cláusula *SELECT* como en el *WHERE*, lo que sugiere que estas técnicas no logran adaptar bien el modelo para esta tarea. Además, presentan un número elevado de SQL inválidos, especialmente Prefix, lo que puede deberse a una falta de estabilidad sintáctica en las secuencias generadas.

El volumen de errores combinados es un buen indicador de cuándo una técnica no logra una representación semántica sólida de la intención original. Técnicas como LoRA y QLoRA reducen significativamente este tipo de errores frente a (IA)³ o Prefix.

4.5.1.4. Análisis de tiempos de inferencia

Además de la precisión en la tarea de conversión NL-to-SQL, también se ha evaluado la eficiencia temporal de cada técnica PEFT aplicada a los modelos. A continuación, se analizan los tiempos de inferencia totales requeridos para procesar las 8.421 muestras del conjunto de evaluación, así como la velocidad de inferencia relativa a lo largo del tiempo.

Tabla 16. *Tiempo total de inferencia por técnica PEFT*

Técnica	Tiempo total (s)
LoRA	515.23
QLoRA	843.61
Prefix	616.54
$(IA)^3$	467.17

Fuente: Elaboración propia.

$(IA)^3$ fue la técnica con el tiempo de inferencia más bajo, completando todo el conjunto en aproximadamente 467 segundos. Esto sugiere que su implementación resulta muy eficiente en términos computacionales, aunque su rendimiento en precisión fue bastante bajo.

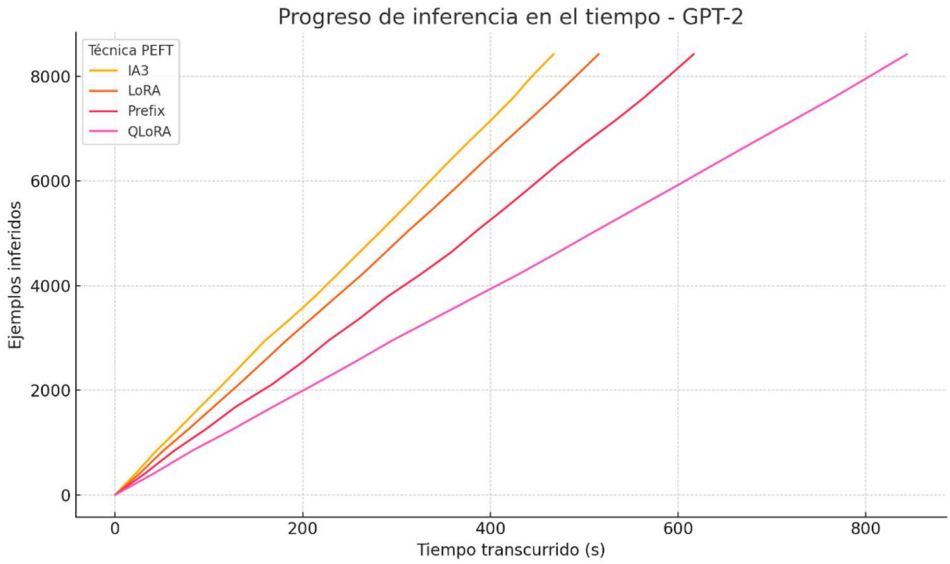
LoRA también demostró buena eficiencia, con un tiempo total de 515 segundos, logrando además los mejores resultados de precisión, lo que la convierte en la técnica más eficaz y eficiente dentro del conjunto evaluado.

Prefix tuning mostró un rendimiento temporal intermedio (617 segundos), aunque sus resultados de precisión fueron los más bajos. Esto indica que no hay una relación directa entre mayor tiempo de inferencia y mejor desempeño para este caso.

QLoRA, a pesar de ser una técnica de cuantización eficiente en en cuanto al uso de memoria, tuvo el tiempo de inferencia más alto (843 segundos). Esto podría deberse a la sobrecarga computacional asociada a la descompresión de pesos cuantizados durante la inferencia, especialmente teniendo en cuenta el uso de hardware no optimizado para este tipo de operaciones.

Finalmente, en la siguiente gráfica se observa el ritmo de inferencia acumulado de cada técnica, mostrando cómo progresan en términos de ejemplos procesados a lo largo del tiempo. Las pendientes más pronunciadas indican una mayor velocidad de procesamiento.

Figura 21. *Ritmo de inferencia acumulado por técnica PEFT*



Fuente: Elaboración propia.

En este sentido, $(IA)^3$ y LoRA muestran pendientes más consistentes y aceleradas, mientras que QLoRA tiene una pendiente más suave, especialmente al principio, cosa que refleja una mayor latencia por cada ejemplo inferido.

4.5.2. Modelo LLAMA-7B

El modelo LLaMA-7B es considerablemente mayor que el modelo GPT-2 (7 vs 1.5 millones de parámetros), lo cual implica una mayor exigencia en términos de consumo de recursos computacionales durante el entrenamiento. La comparativa realizada muestra variaciones importantes en tiempo de entrenamiento, velocidad de ejecución de pasos y utilización de memoria GPU según la técnica PEFT empleada.

4.5.2.1. Análisis de rendimiento con recursos limitados (Objetivo 1)

En primer lugar, en términos de tiempo total de entrenamiento, se destaca claramente la eficiencia relativa de las técnicas LoRA y QLoRA, con tiempos de entrenamiento de entorno a 5-6 horas, frente a $(IA)^3$ (32 horas) y especialmente Prefix Tuning (57 horas). El notable

incremento del tiempo en estas últimas técnicas refleja diferencias fundamentales en su metodología de ajuste fino. En particular, Prefix Tuning implica ajustar y aprender prefijos adicionales que requieren un procesamiento significativamente más extenso, provocando así tiempos de entrenamiento extremadamente largos. $(IA)^3$, aunque menos extrema que Prefix Tuning, sigue requiriendo muchas más horas para ajustar sus parámetros escalares individuales.

Respecto a la velocidad de ejecución de pasos por segundo, LoRA (1.879 steps/s) y QLoRA (1.491 steps/s) nuevamente destacan como técnicas significativamente más rápidas que $(IA)^3$ y Prefix Tuning (0.302 y 0.170 steps/s respectivamente). Estas diferencias están directamente relacionadas con la cantidad y naturaleza de los parámetros ajustados. LoRA y QLoRA modifican menos parámetros en cada paso, incrementando así la velocidad del entrenamiento.

Tabla 17. *Comparativa de tiempos y uso de recursos (LLAMA-7B)*

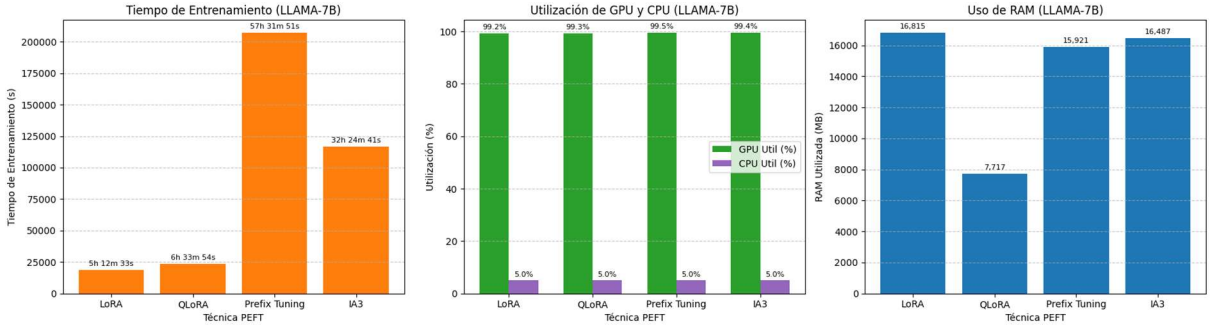
Técnica	Train Time	Train Steps/s	GPU Mem (MB)	GPU Util (%)	CPU Util (%)
LoRA	05:12:33	1.879	16815	99.19	5.0
QLoRA	06:33:54	1.491	7718	99.29	5.0
Prefix	57:31:51	0.170	15922	99.47	5.0
$(IA)^3$	32:24:41	0.302	16488	99.41	5.0

Fuente: Elaboración propia.

En cuanto al consumo de memoria GPU, se observa un patrón notablemente favorable en QLoRA, que utiliza cuantización de pesos para reducir el consumo de memoria (7718 MB), menos de la mitad del consumo requerido por LoRA, $(IA)^3$ y Prefix Tuning. Este aspecto puede ser crucial en entornos con restricciones severas de memoria GPU, mostrando a QLoRA como una opción muy atractiva en situaciones donde la memoria es limitada.

Por otro lado, todas las técnicas muestran una utilización muy alta de GPU, situada alrededor del 99%, lo que refleja una utilización eficiente y casi máxima de los recursos gráficos disponibles. La utilización de CPU permanece constante alrededor del 5% para todas las técnicas, indicando claramente que la limitación fundamental no está en el procesamiento CPU.

Figura 22. *Comparativa uso de recursos técnicas PEFT sobre el modelo LLAMA-7B*



Fuente: Elaboración propia.

QLoRA sobresale en consumo eficiente de memoria GPU, siendo claramente la técnica recomendada para el ajuste del modelo LLAMA-7B, aunque sacrificando ligeramente la velocidad frente a LoRA.

LoRA presenta el mejor equilibrio general entre tiempo de entrenamiento y rendimiento.

$(IA)^3$ y especialmente Prefix Tuning se presentan como técnicas con un coste temporal y computacional muy elevado, lo que las hace menos atractivas.

4.5.2.2. Análisis de precisión en la tarea NL-to-SQL (Objetivo 2)

En lo que a valores de pérdida se refiere, la técnica QLoRA obtiene los mejores resultados, seguida muy de cerca por LoRA. Ambas técnicas muestran valores bajos tanto en entrenamiento como en evaluación, evitando el sobreajuste y confirmando un buen rendimiento tanto en generalización como en el ajuste inicial sobre el conjunto de entrenamiento.

La técnica Prefix Tuning, por el contrario, muestra un rendimiento inferior con unos valores de pérdida bastante más altos, tanto en entrenamiento como en evaluación. Esto sugiere que, aunque esta técnica introduce un mecanismo flexible mediante prefijos adicionales de contexto, necesita ajustes más profundos en hiperparámetros o más épocas para converger adecuadamente, lo que aumenta considerablemente el tiempo necesario para alcanzar un rendimiento competitivo.

$(IA)^3$, con un rendimiento aceptable pero menos destacado está en un punto intermedio. Aunque mejor que Prefix Tuning, su rendimiento sigue siendo inferior a LoRA y QLoRA. Aunque es capaz de ajustar el modelo razonablemente bien, no alcanza los niveles de

eficiencia en términos de generalización logrados por LoRA y QLoRA, especialmente dado su coste temporal más elevado.

En lo referente a la magnitud promedio de los gradientes, nuevamente LoRA muestra valores más altos, que se traducen en ajustes más agresivos y rápidos durante el entrenamiento, potencialmente facilitando una convergencia eficiente. QLoRA presenta gradientes más moderados, pero capaces de lograr una buena convergencia y generalización con menor consumo de memoria GPU. Para Prefix Tuning e $(IA)^3$ se observan gradientes considerablemente menores, lo que se traduciría en ajustes más sutiles, potencialmente menos efectivos para una rápida convergencia y generalización.

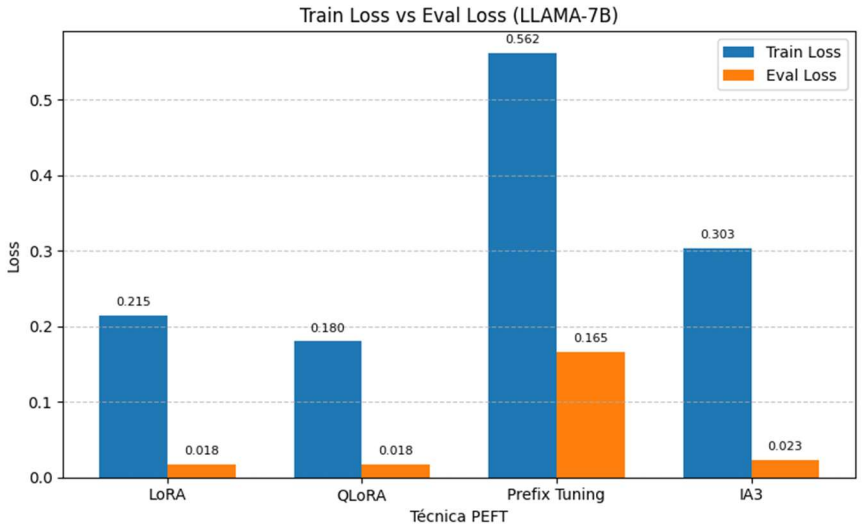
Las tasas de aprendizaje seleccionadas por defecto para LoRA y QLoRA son considerablemente mayores, facilitando una convergencia más rápida. Prefix e $(IA)^3$ utilizan unos valores mucho menores, requiriendo por tanto muchas más épocas y recursos para alcanzar resultados comparables.

Tabla 18. *Métricas de entrenamiento por técnica PEFT sobre el modelo LLaMA-7B*

Técnica	Epoch	Train Loss	Eval Loss	Grad Norm	Learning Rate
LoRA	0.88	0.2145	0.01776	0.4066	4.5607e-05
QLoRA	0.88	0.1800	0.01766	0.1813	4.5748e-05
Prefix	9.99	0.5622	0.16549	0.08152	6.2447e-08
$(IA)^3$	5.54	0.3031	0.02343	0.02945	2.2333e-05

Fuente: Elaboración propia.

Figura 23. *Comparativa función de pérdida técnicas PEFT sobre el modelo LLaMA-7B*



Fuente: Elaboración propia.

QLoRA ofrece el mejor balance entre uso de recursos y rendimiento, destacando por su bajo consumo de memoria GPU y excelente capacidad de generalización.

Prefix Tuning requiere mejoras significativas en hiperparámetros o una cantidad de entrenamiento considerablemente mayor para alcanzar resultados competitivos, haciendo difícil justificar su elevado coste temporal y computacional sin ajustes profundos adicionales.

En definitiva, en entornos limitados en recursos computacionales, QLoRA y LoRA son las técnicas claramente ganadoras, proporcionando resultados óptimos tanto en eficiencia de entrenamiento como en calidad de traducción natural a SQL para el modelo LLaMA-7B.

4.5.2.3. Resultados de precisión en inferencia

Al igual que para el modelo anterior, en este análisis de precisión en inferencia se utilizó el conjunto de evaluación de WikiSQL, evaluando la calidad de las consultas SQL generadas por el modelo LLaMA-7B.

Las siguientes figuras resumen la distribución de los resultados por técnica aplicada sobre LLaMA-7B:

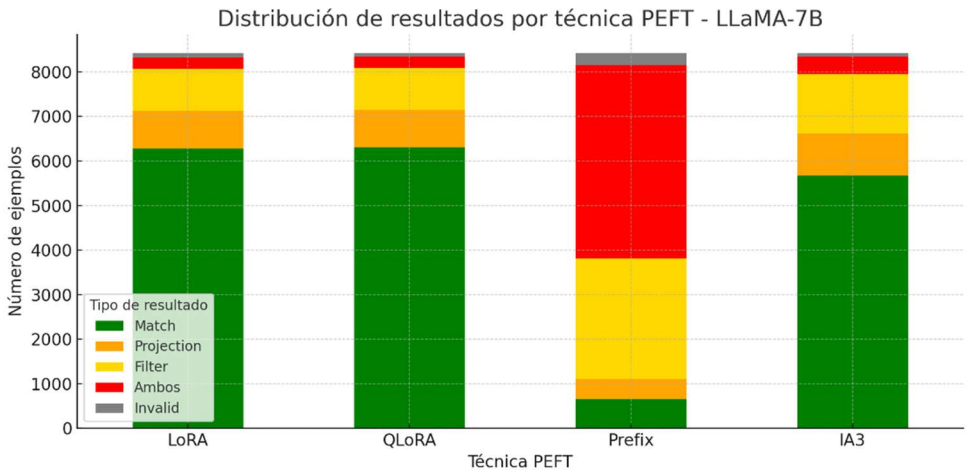
Tabla 19. *Distribución de resultados por técnica PEFT (LLAMA-7B)*

Técnica PEFT	Match	Projection	Filter	Ambos	Invalid
LoRA	6277	851	938	256	99
QLoRA	6304	847	939	248	83

Prefix	659	446	2706	4341	269
(IA)³	5668	947	1338	393	75

Fuente: Elaboración propia.

Figura 24. *Distribución de resultados por técnica PEFT (LLaMA-7B)*



Fuente: Elaboración propia.

LoRA y QLoRA destacan por alcanzar más de 74% de aciertos exactos, demostrando que LLaMA-7B afinado con estas técnicas, es altamente competente en la tarea de generación de consultas SQL correctas. En ambas técnicas, los errores se distribuyen en proporciones similares: ligeras imprecisiones en la proyección (*SELECT*) y en el filtrado (*WHERE*), con una baja incidencia de errores combinados y de consultas inválidas.

(IA)³ también alcanza un rendimiento alto con 66% de aciertos, pero con más errores en proyecciones y filtros en comparación con LoRA/QLoRA. La cantidad de errores combinados y de SQL inválidas, sin embargo, se mantiene baja, lo que indica cierta robustez semántica en la generación.

Por su lado, Prefix tuning, presenta un comportamiento deficiente, con solo 659 coincidencias exactas (menos de un 8%). Más de 5.000 errores combinados y 2.700 errores de filtrado, lo cual nos indica que esta técnica no logra adaptar eficazmente el modelo LLaMA-7B a la tarea NL-to-SQL. Además, generó la mayor cantidad de consultas inválidas (269), reflejando problemas sintácticos persistentes.

4.5.2.4. Análisis de tiempos de inferencia

En esta sección se analizan los tiempos de inferencia totales requeridos para procesar el conjunto de muestras de evaluación de WikiSQL, así como la velocidad de inferencia relativa a lo largo del tiempo.

Tabla 20. *Tiempo total de inferencia por técnica PEFT sobre LLAMA-7B*

Técnica	Tiempo total (s)
LoRA	9418.78
QLoRA	5316.54
Prefix	9312.31
$(IA)^3$	9076.35

Fuente: Elaboración propia.

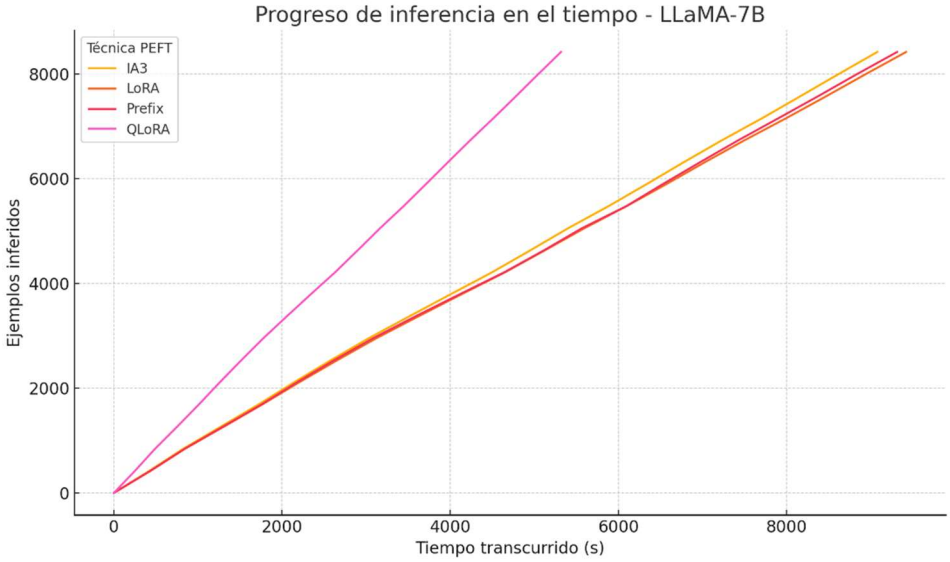
En este escenario, QLoRA es la técnica con menor tiempo de inferencia, completando el conjunto en unos 5.316 segundos. Esto podría deberse a la eficiencia de representación que permite su mecanismo de cuantización, siempre que el entorno de ejecución esté optimizado para dicha operación.

Las otras tres técnicas presentan tiempos similares, todos superiores a los 9.000 segundos, lo que indica una mayor carga computacional, posiblemente debido al tamaño del modelo y a operaciones menos optimizadas para una inferencia a gran escala.

A pesar de que Prefix tuning, como ya se vió anteriormente, mostró un desempeño muy pobre en precisión, su tiempo de inferencia fue casi idéntico al de LoRA o $(IA)^3$. Esto confirma que un mayor tiempo de ejecución no implica necesariamente mejores resultados, sino que depende de cómo se integran los parámetros adicionales en el modelo base.

La siguiente gráfica muestra el ritmo de inferencia acumulado de cada técnica a lo largo del tiempo, donde pendientes pronunciadas indican mayor velocidad de procesamiento.

Figura 25. *Progreso temporal de inferencia (LLaMA-7B)*



Fuente: Elaboración propia.

En este sentido y, al contrario de lo que sucedía en el modelo GPT-2, QLoRA muestra una pendiente más pronunciada, reflejando una mayor velocidad de inferencia.

En cambio, LoRA, $(IA)^3$ y Prefix muestran curvas similares, más suaves, con un ritmo de inferencia relativamente constante pero más lento.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

En el presente trabajo de fin de estudios se ha abordado el problema de cómo aplicar técnicas de ajuste fino eficiente en parámetros a modelos LLM para la generación de consultas SQL a partir de lenguaje natural en entornos con recursos computacionales limitados. Este reto se ha afrontado mediante un estudio comparativo de distintas técnicas, incluyendo LoRA, QLoRA, Prefix Tuning e $(IA)^3$, sobre modelos base como GPT-2 y LLaMA-7B, si bien no fue posible implementar ni analizar la técnica BitFit debido a incompatibilidades con la versión de los modelos y a la ausencia de soporte estable en las librerías utilizadas. La metodología seguida ha permitido evaluar de forma sistemática el consumo de recursos, el rendimiento y la capacidad de generalización de cada técnica, aunque las restricciones temporales y el coste económico de contratar entornos de ejecución en la nube han limitado la exploración detallada de los hiperparámetros que potencialmente habrían podido optimizar los resultados obtenidos.

La solución propuesta se basa en un análisis experimental que demuestra la viabilidad de técnicas como QLoRA y LoRA para entornos con limitaciones de hardware, gracias a su bajo consumo de memoria y rapidez de convergencia. No obstante, el uso exclusivo de un conjunto de datos en inglés supone una limitación que aconseja explorar en trabajos futuros datasets multilingües que permitan evaluar la aplicabilidad de estos enfoques en otros idiomas.

Las principales contribuciones del trabajo incluyen la recopilación y comparación rigurosa de métricas de eficiencia y precisión, la documentación detallada del proceso de ajuste fino y la validación empírica de los resultados. Estas aportaciones se relacionan de manera directa con los objetivos establecidos: identificar las técnicas más adecuadas para escenarios con recursos limitados y valorar su aplicabilidad en la tarea concreta de traducción NL-to-SQL. En este sentido, se puede concluir que los objetivos han sido alcanzados en gran medida, al proporcionar evidencia sólida sobre qué enfoques resultan más recomendables en diferentes escenarios.

En conjunto, el trabajo aporta una referencia práctica y actualizada que puede ser de utilidad tanto para el ámbito académico como profesional, sirviendo de guía a futuros proyectos que

requieran optimizar modelos LLM en condiciones de restricciones computacionales, al tiempo que plantea oportunidades de mejora y ampliación en futuros estudios.

5.2. Líneas de trabajo futuro

A partir de los resultados obtenidos en este estudio y de las limitaciones identificadas tanto en la complejidad de las consultas SQL generadas como en las restricciones computacionales del entorno experimental, se proponen a continuación diversas líneas de trabajo futuro estructuradas en **dos ejes principales de investigación**:

1. **Ampliación en la complejidad de las tareas NL-to-SQL**, con el objetivo de evaluar y mejorar la capacidad de los modelos LLM para generar consultas SQL más sofisticadas, que reflejen de forma más realista los desafíos presentes en entornos de bases de datos relacionales con múltiples tablas, relaciones jerárquicas y operaciones avanzadas.
2. **Optimización del rendimiento en escenarios con recursos computacionales limitados**, profundizando en el uso de técnicas PEFT adaptadas a hardware de baja capacidad, con el fin de mejorar la eficiencia sin comprometer la precisión del modelo.

Cada una de estas líneas responde a necesidades complementarias: por un lado, **incrementar la dificultad del problema para asegurar la robustez y escalabilidad de los modelos**, y por otro, **reducir la carga computacional para favorecer su aplicabilidad en entornos reales con restricciones tecnológicas o presupuestarias**. A continuación, se detallan propuestas específicas para cada una de estas dos líneas de investigación.

5.2.1. Ampliación en complejidad de NL-to-SQL

1. Uso de datasets con estructuras multitabla y consultas complejas

Explorar el uso de datasets más avanzados que incluyan relaciones entre múltiples tablas, como:

- **Spider**: incluye consultas con *JOINs*, *nested queries*, *GROUP BY*, *ORDER BY* y subconsultas.
- **WikiSQL-S** o **SParC**, orientados a diálogos y consultas secuenciales complejas. Estos conjuntos permiten entrenar y evaluar modelos que desarrollan un entendimiento más profundo de la semántica relacional y lógica de bases de datos reales

2. **Incorporación de esquemas de base de datos explícitos durante el entrenamiento**

Investigar cómo los modelos pueden utilizar representaciones explícitas del esquema (por ejemplo, diagramas de relaciones o JSON de metadatos) para mejorar la capacidad de inferir relaciones entre tablas y construir JOINs correctos. Esto también incluye el aprendizaje de restricciones como claves primarias/foráneas.

3. **Fine-tuning multitarea con generación condicional estructurada**

Diseñar experimentos de fine-tuning donde el modelo aprenda no solo a generar la consulta SQL, sino también a identificar las tablas relevantes y predecir las relaciones necesarias antes de construir la query. Esta descomposición puede facilitar la generalización hacia estructuras más complejas.

4. **Evaluación progresiva por niveles de dificultad SQL**

Incorporar una métrica o esquema de evaluación que clasifique las consultas según su dificultad estructural (basado en el número de *JOIN*s, condiciones anidadas, agregaciones, etc.), y estudiar el rendimiento de cada técnica PEFT según esta progresión. Esto permitiría entender mejor los límites de cada técnica.

5. **Entrenamiento con explicaciones semánticas supervisadas**

Explorar datasets en los que las consultas SQL estén acompañadas de explicaciones en lenguaje natural que describan su estructura lógica (por ejemplo, "esta consulta selecciona los nombres de los clientes que han hecho pedidos por más de 100€, uniendo la tabla de clientes con pedidos…"). Esta información adicional puede reforzar el aprendizaje estructurado.

5.2.2. Optimización del rendimiento en distintos escenarios

1. **Exploración de combinaciones híbridas de técnicas PEFT**

Investigar el impacto de aplicar múltiples técnicas PEFT de forma combinada (por ejemplo, QLoRA con Prefix Tuning o BitFit con $(IA)^3$), evaluando sinergias y mejoras en eficiencia o precisión. Estas combinaciones podrían aprovechar los puntos fuertes de cada técnica, especialmente en tareas complejas como NL-to-SQL.

2. **Integración con técnicas de recuperación de información (RAG)**

Aunque el enfoque RAG fue excluido del alcance de este estudio por centrarse en la inferencia, futuras investigaciones podrían explorar cómo combinar PEFT con RAG para potenciar la precisión semántica y el acceso a conocimiento actualizado, especialmente en contextos corporativos o educativos.

3. **Comparativa en hardware extremadamente restringido (edge computing)**

Evaluar el rendimiento de las técnicas PEFT en entornos aún más limitados, como dispositivos móviles o hardware embebido, sería útil para determinar su viabilidad en escenarios de edge AI, donde los recursos son especialmente escasos.

4. **Evaluación de técnicas emergentes y adaptativas (e.g., AdaLoRA, ReLoRA)**

Incorporar nuevas técnicas PEFT que continúan surgiendo en el estado del arte, como AdaLoRA o ReLoRA, y evaluar su impacto tanto en rendimiento como en estabilidad del fine-tuning bajo restricciones de memoria y tiempo.

5. **Optimización de hiperparámetros y estrategias de entrenamiento eficientes**

Algunas técnicas PEFT requieren una calibración precisa para obtener buenos resultados. El diseño de estrategias automáticas de ajuste de hiperparámetros o el uso de metaaprendizaje podría reducir la dependencia de la experimentación manual intensiva.

Referencias bibliográficas

Ben Zaken, E., Ravfogel, S., & Goldberg, Y. (2021). BitFit: Simple Parameter-Efficient Fine-Tuning for Transformer-Based Masked Language Models. *ArXiv Preprint ArXiv:2106.10199*. https://arxiv.org/abs/2106.10199

Ben Zaken, E., Ravfogel, S., & Goldberg, Y. (2023). Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning. *ArXiv Preprint ArXiv:2303.15647*. https://arxiv.org/pdf/2303.15647.pdf

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., & Amodei, D. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems* (Vol. 33, pp. 1877–1901).

Clark, K., Luong, M.-T., Le, Q. V, & Manning, C. D. (2020). ELECTRA: Pre-training text encoders as discriminators rather than generators. *International Conference on Learning Representations (ICLR 2020)*. https://openreview.net/forum?id=r1xMH1BtvB

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/2305.14314

Fu, C.-L., Chen, Z.-C., Lee, Y.-R., & Lee, H. (2022). *AdapterBias: Parameter-efficient Token-dependent Representation Shift for Adapters in NLP Tasks*. https://arxiv.org/abs/2205.00305

Howard, J., & Ruder, S. (2018). Universal Language Model Fine-tuning for Text Classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, 328–339. https://aclanthology.org/P18-1031/

Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *ArXiv Preprint ArXiv:2106.09685*. https://arxiv.org/abs/2106.09685

Hugging Face. (2022). *Transformers Documentation*. https://huggingface.co/docs/transformers/index

Jawade, B. (2023). *Understanding LoRA: Low-Rank Adaptation for Finetuning Large Models*. https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6

Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., & Liu, Q. (2020). TinyBERT: Distilling BERT for natural language understanding. *Findings of EMNLP 2020*, 4163–4174. https://doi.org/10.18653/v1/2020.findings-emnlp.372

Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The efficient transformer. *International Conference on Learning Representations (ICLR 2020)*. https://arxiv.org/abs/2001.04451

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv Preprint ArXiv:1909.11942*. https://arxiv.org/abs/1909.11942

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *ArXiv Preprint ArXiv:2005.11401*. https://arxiv.org/abs/2005.11401

Li, X. L., & Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. *ArXiv Preprint ArXiv:2101.00190*. https://arxiv.org/abs/2101.00190

Lialin, V., Muckatira, S., Shivagunde, N., & Rumshisky, A. (2023). ReLoRA: High-Rank Training Through Low-Rank Updates. *ArXiv Preprint ArXiv:2307.05695*. https://arxiv.org/abs/2307.05695

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., & Raffel, C. (2022). Few-shot Parameter-Efficient Fine-tuning is Better and Cheaper than In-Context Learning. *ArXiv Preprint ArXiv:2205.05638*. https://arxiv.org/abs/2205.05638

Liu, X., Ji, Q., Fu, Y., Tam, D., Ji, H., Chang, K.-W., & Han, X. (2021a). Pre-train Prompt Tune: Towards Generalizable Few-shot Conditional Text Generation. *ArXiv Preprint ArXiv:2103.10385*. https://arxiv.org/abs/2103.10385

Liu, X., Ji, Q., Fu, Y., Tam, D., Ji, H., Chang, K.-W., & Han, X. (2021b). P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks. *ArXiv Preprint ArXiv:2110.07602*. https://arxiv.org/abs/2110.07602

Lv, K., Yang, Y., Liu, T., Gao, Q., Guo, Q., & Qiu, X. (2023). Full parameter fine-tuning for large language models with limited resources. *ArXiv Preprint ArXiv:2306.09782*. https://arxiv.org/pdf/2306.09782.pdf

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). *Large Language Models: A Survey*. https://arxiv.org/abs/2402.06196

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). *A Comprehensive Overview of Large Language Models*. https://arxiv.org/abs/2307.06435

Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., & Gurevych, I. (2021). AdapterFusion: Non-Destructive Task Composition for Transfer Learning. *ArXiv Preprint ArXiv:2005.00247*. https://arxiv.org/abs/2005.00247

Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., Fox, E., & Larochelle, H. (2021). Improving Reproducibility in Machine Learning Research: A Report from the NeurIPS 2019 Reproducibility Program. *Journal of Machine Learning Research*, *22*(167), 1–20.

Quamar, A., Efthymiou, V., Lei, C., & Ozcan, F. (2022). *Natural Language Interfaces to Data*. https://arxiv.org/abs/2212.13074

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language models are unsupervised multitask learners*. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, *21*(140), 1–67. http://jmlr.org/papers/v21/20-074.html

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv Preprint ArXiv:1910.01108*. https://arxiv.org/abs/1910.01108

Shearer, C. (2000). The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing*, *5*(4), 13–22. https://mineracaodedados.files.wordpress.com/2012/04/the-crisp-dm-model-the-new-blueprint-for-data-mining-shearer-colin.pdf

Sun, Z., Yu, H., Song, X., Liu, R., Yang, Y., & Zhou, D. (2020). MobileBERT: A compact task-agnostic BERT for resource-limited devices. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, 2158–2170. https://doi.org/10.18653/v1/2020.acl-main.195

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and efficient foundation language models. *ArXiv Preprint ArXiv:2302.13971*. https://arxiv.org/abs/2302.13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All You Need. *ArXiv Preprint ArXiv:1706.03762*. https://arxiv.org/abs/1706.03762

Verma, P., Vijay, D., Panwar, S., Mehta, M., Patel, K., & Naidu, A. (2024). The ultimate guide to fine-tuning LLMs from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *ArXiv Preprint ArXiv:2408.13296*. https://arxiv.org/pdf/2408.13296.pdf

Weng, B. (2024). Navigating the Landscape of Large Language Models: A Comprehensive Review and Analysis of Paradigms and Fine-Tuning Strategies. *ArXiv Preprint ArXiv:2404.09022*. https://arxiv.org/pdf/2404.09022.pdf

Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., & Wang, F. L. (2023). Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment. *ArXiv Preprint ArXiv:2312.12148*. https://arxiv.org/abs/2312.12148

Yang, H., Zhang, Y., Xu, J., Lu, H., Heng, P. A., & Lam, W. (2024). Unveiling the Generalization Power of Fine-Tuned Large Language Models. *ArXiv Preprint ArXiv:2403.09162*. https://arxiv.org/pdf/2403.09162.pdf

Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., Chen, W., & Zhao, T. (2023). AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning.

International Conference on Learning Representations (ICLR).

https://arxiv.org/abs/2303.10512

Zhong, V., Xiong, C., & Socher, R. (2017a). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*, *abs/1709.00103*.

Zhong, V., Xiong, C., & Socher, R. (2017b). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*, *abs/1709.00103*.

Zhu, L., Hu, L., Lin, J., & Han, S. (2023). LIFT: Efficient layer-wise fine-tuning for large model models. *ArXiv Preprint ArXiv:2306.07947*. https://arxiv.org/pdf/2306.07947.pdf

Anexo A. Código fuente y datos analizados

A continuación, se incluye la referencia al repositorio GitHub donde se ha publicado el código fuente completo utilizado en el desarrollo de este trabajo de fin de máster. En dicho repositorio se encuentran todos los scripts, configuraciones y recursos necesarios para reproducir los experimentos de fine-tuning, así como las instrucciones detalladas para su puesta en marcha.

- **https://github.com/toni-carrasco/fine-tune-image**

El enlace proporciona acceso al código completo de los módulos de pre-procesamiento, los scripts de análisis de datos, el código de entrenamiento con distintas técnicas PEFT y los ficheros de configuración necesarios para replicar los resultados descritos en esta memoria. Asimismo, se incluyen ejemplos de uso, documentación adicional y una guía paso a paso para facilitar la instalación y ejecución en entornos con recursos limitados.

Anexo B. Costes de contratación del entorno de pruebas en la nube