ISAAC BERMUDEZ PUENTES

Tesis_Isaac_Bermúdez_Puentes

Caja 2

Detalles del documento

Identificador de la entrega
trn:oid:::26268:603805258

Fecha de entrega
30 jun 2026, 12:33 p.m. GMT-5

Fecha de descarga
30 jun 2026, 12:40 p.m. GMT-5

Nombre del archivo
Tesis_Isaac_Bermúdez_Puentes.pdf

Tamaño del archivo
1.4 MB

79 páginas

17.655 palabras

93.196 caracteres

7% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Coincidencias menores (menos de 12 palabras)

Fuentes principales

6% Fuentes de Internet
4% Publicaciones
7% Trabajos entregados (trabajos del estudiante)

Fuentes principales

6% Fuentes de Internet
4% Publicaciones
7% Trabajos entregados (trabajos del estudiante)

Fuentes principales

Las fuentes con el mayor número de coincidencias dentro de la entrega. Las fuentes superpuestas no se mostrarán.

1 Internet
pdfcoffee.com <1%

2 Trabajos del estudiante
Uniminuto Virtual on 2026-03-27 <1%

3 Trabajos del estudiante
University of Greenwich on 2024-11-19 <1%

4 Internet
arxiv.org <1%

5 Publicación
Juan Carlos Carmona, Raúl Atienza, Raúl Redondo, José R. Iribarren. "Chapter 38 ... <1%

6 Trabajos del estudiante
University of West Attica on 2025-06-22 <1%

7 Internet
osuva.uwasa.fi <1%

8 Publicación
Yifan Shen, Chunyan Li. "Design and Implementation of an Intelligent Question A... <1%

9 Trabajos del estudiante
University of Greenwich on 2024-04-08 <1%

10 Trabajos del estudiante
University of Greenwich on 2024-04-08 <1%

11 Trabajos del estudiante
University of the Andes on 2024-03-12 <1%

12 Internet
eiusc.eiu.edu.vn <1%

13 Internet
repository.udistrital.edu.co <1%

14 Internet
eprints.whiterose.ac.uk <1%

15 Trabajos del estudiante
College of Professional and Continuing Education (CPCE), Polytechnic University o... <1%

16 Trabajos del estudiante
Universidad Internacional de la Rioja on 2025-03-10 <1%

17 Internet
www.frontiersin.org <1%

18 Internet
www.asobancaria.com <1%

19 Trabajos del estudiante
Universidad Anahuac México Sur on 2024-05-11 <1%

20 Internet
smartin.tecnm.mx <1%

21 Internet
www.revistasinvestigacion.esic.edu <1%

22 Trabajos del estudiante
University of the Sunshine Coast on 2023-06-02 <1%

23 Trabajos del estudiante
George Mason University on 2026-03-02 <1%

24 Trabajos del estudiante
Universidad de San Buenaventura on 2020-04-09 <1%

25 Internet
alcaldiasoacha.gov.co <1%

26 Trabajos del estudiante
National University College - Online on 2025-01-18 <1%

27 Publicación
Allan Herminio Vargas Garcia. "La banca digital: Innovación tecnológica en la incl... <1%

28 Internet
www.ibm.com <1%

29 Trabajos del estudiante
Sophia University on 2025-08-05 <1%

30 Trabajos del estudiante
UNIBA on 2024-10-29 <1%

31 Trabajos del estudiante
Universidad TecMilenio on 2024-02-13 <1%

32 Trabajos del estudiante
Universitat Oberta de Catalunya on 2026-06-02 <1%

33 Internet
riujap.ujap.edu.ve <1%

34 Internet
www.corporaciongilbertoecheverri.gov.co <1%

35 Trabajos del estudiante
Houston Community College on 2025-07-31 <1%

36 Trabajos del estudiante
UNIBA on 2026-03-02 <1%

37 Trabajos del estudiante
Universidad Autónoma de Nuevo León on 2024-03-05 <1%

38 Trabajos del estudiante
Universidad Internacional de la Rioja on 2024-01-19 <1%

39 Trabajos del estudiante
University of Greenwich on 2024-04-05 <1%

40 Internet

certidevs.com <1%

41 Internet

institutojubones.edu.ec <1%

42 Internet

ojs.aaai.org <1%

43 Internet

www.coursehero.com <1%

44 Trabajos del estudiante

University of New South Wales on 2014-06-03 <1%

45 Internet

dokumen.pub <1%

46 Internet

empiezoinformatica.wordpress.com <1%

47 Internet

vitela.javerianacali.edu.co <1%

Implementación de un modelo de procesamiento de lenguaje natural para facilitar el uso de la información financiera sobre tasas de interés en créditos de Colombia

Isaac Bermúdez Puentes

Pontificia Universidad Javeriana

Facultad de Ciencias

Pregrado en Ciencia de datos

Dirección:

Andrea Del Pilar Rueda Olarte y

Miguel Ángel Méndez Hernández

Bogotá D.C

2026

Tabla de contenido

1. Introducción 5
2. Planteamiento del problema 7
3. Justificación 8
4. Objetivos 9
Objetivo General 9
Objetivos específicos 9
5. Marco Teórico 10
5.1 Gobierno de datos abiertos 10
5.2 Inclusión financiera 11
5.3 Procesamiento de lenguaje natural (PNL) 12
5.4 Campo del *Text to SQL* 14
5.5 Redes Neuronales 15
5.6 Arquitectura Transformer autoregresivo 16
5.6.1 Word Embeddings y RoPE 16
5.6.2 Self attention causal 18
5.6.3 Multi-head attention 19
5.6.4 Grouped query attention 21
5.6.5 Sliding window attention 22
5.6.6 Feed forward network 22
5.7 Técnica supervisada de Fine-tunig LoRA 23
6. Metodología 25
7. Cronograma 32
8. Desarrollo de la metodología 33
8.1 Exploración inicial y *benchmarking* 33
8.1.1 Exploración preliminar 33
8.1.2 Exploración inicial y elección de modelos 35
8.2 Preparación y procesamiento de datos 43
8.3 desarrollo e implementación de modelos en el contexto problema 50
8.4 Fine-tunning 50
8.5 Exportación de los modelos 53
8.6 Presentación de la interfaz 54

9. Resultados..... 57
10. Conclusiones..... 63
11. Referencias..... 66
12. Apéndice 68

turnitin Página 9 de 85 - Engrega de integridad Identificador de la entrega trn:oid:::26268:603805258

Resumen

En Colombia, a pesar de la creciente era digital y la masificación de fuentes de consulta inmediatas, la brecha de desinformación sobre el área financiera emitida por los entes gubernamentales sigue sin verse afectada. Una de las posibles causas es la forma en la que se presentan los datos dentro del portal de datos abiertos, debido a que, para personas sin conocimientos técnicos en manejo de datos, es de difícil acceso la información que podrían transformar en valor. De acuerdo con esto, se desarrolló una herramienta que facilita el uso de los datos en el ámbito de las tasas de interés activas, información clasificada dentro de la sección economía y finanzas, por medio de la implementación de un LLM con *fine-tuning* de tipo LoRA que realiza la tarea de traducción de consultas en lenguaje natural a SQL. Esta labor se desarrolló a través de una metodología de trabajo CRISP-DM, dividida en fases como exploración, preparación y limpieza de datos, desarrollo y evaluación de resultados. De lo encontrado, se evidencia cómo el modelo de Mistral con fine-tuning LoRA tiene comparablemente mejor desempeño que SQLCoder; en consecuencia, se utilizó la configuración con mejores métricas para implementarse en la interfaz. Por último, en términos generales el proyecto se realizó con retos que debieron superarse, como las limitaciones computacionales. Se reconoce la herramienta como un producto eficiente y adecuado para contrarrestar el problema planteado, así como se presentan ideas de mejora como el incremento de muestras y se manifiestan los posibles caminos a seguir para implementarla en nuevos ámbitos.

Abstract

In Colombia, despite the growing digital era and the widespread availability of immediate information sources, the misinformation gap regarding financial data published by government entities remains unaffected. One of the possible causes is the way in which data is presented within the open data portal, since for individuals without technical knowledge in data management, the information that could be transformed into value is difficult to access. Accordingly, a tool was developed to facilitate the use of data in the field of active interest rates — information classified under the economy and finance section — through the implementation of an LLM with LoRA fine-tuning, which performs the task of translating natural language queries into SQL. This work was carried out using the CRISP-DM methodology, divided into phases such as data exploration, preparation and cleaning, development, and evaluation of results. The findings show that the Mistral model with LoRA fine-tuning achieves comparably better performance than SQLcoder; consequently, the configuration with the best metrics was selected for implementation in the interface. Finally, in general terms, the project was carried out amid challenges that had to be overcome, such as computational limitations. The tool is recognized as an efficient and suitable product to address the problem stated, while also presenting improvement ideas such as increasing sample sizes and outlining possible future paths for its implementation in new domains.

1. Introducción

El foco principal de la presente investigación se basa en facilitar el uso de los datos que están compartidos en la plataforma de datos abiertos Colombia dentro del sector financiero en temas de tasas de interés activas, por medio de un modelo de lenguaje de inteligencia artificial para el procesamiento de texto, el cual realiza una traducción automática a lenguaje de consulta estructurada (*structured query language*, SQL), para la interacción con datos en formato tabular.

Este proyecto nace de la necesidad de poder explotar el gran potencial de información que reposa en los repositorios gubernamentales que muchas veces no es posible consultar de manera sencilla y eficiente, el formato en el cual se encuentra resulta ser técnico o poco amigable para las personas del común que podrían estar haciendo uso de esta y generando un valor agregado en su toma de decisiones informada, por lo cual el desarrollo de esta herramienta genera un impacto positivo sobre la brecha de desinformación que está presente al momento de poder consultar las solicitudes que los usuarios se plantean. Adicionalmente se resalta la importancia del uso de métodos y modelos relacionados con la inteligencia artificial como una respuesta al progreso inminente que ha tenido el campo en tiempos recientes, su implementación genera una nueva forma de encarar el problema de falta de información con una solución certera y escalable a diferentes fuentes de información.

La ruta que se desarrollara para poder cumplir con los diferentes objetivos establecidos será la exploración de literatura y diferentes modelos para poder seleccionar el que presenta

mejor comportamiento, posteriormente la recopilación, procesamiento y almacenamiento de la fuente de información de datos abiertos Colombia, diseño de fuente de validación y entrenamiento o pares (datos con preguntas en lenguaje natural y su respectiva sentencia SQL) , implementación de técnica supervisada de *fine-tunig* sobre un modelo de lenguaje a gran escala *(Large Language Model, LLM)* preentrenado, el desarrollo de la interfaz se centrara en un diseño sencillo, garantizando la usabilidad.

La herramienta por diseñar será una interfaz donde usuarios puedan escribir una pregunta en relación con la temática de tasas de interés e información asociada y puedan obtener una respuesta confiable a partir de los datos recopilados del portal de datos abiertos Colombia.

2. Planteamiento del problema

La transmisión de información pública por medio de los entes gubernamentales en plataformas digitales es una estrategia para garantizar un gobierno con transparencia y responsabilidad, impulsando la investigación a través de la información pública. Estas acciones están respaldadas por normas y leyes que garantizan la reutilización y disposición de la información a cualquier persona, sin embargo; cuando se habla de accesibilidad no se tiene en cuenta la usabilidad, compresión y aprovechamiento por parte de usuarios no expertos. Como resultado se ve reflejada una brecha de desinformación y por lo tanto un problema antidemocrático sobre los datos abiertos, pues solo personas con el suficiente conocimiento practico para analizar y procesar los datos tabulares compartidos pueden hacer uso de este. Reforzando esta posición según Zuiderwijk et al. (2016), al igual que en Colombia en más países como el Reino Unido, el portal de datos abiertos presenta la información en formatos tabulares o técnicos que, si bien son legibles por máquina, no resultan prácticos para usuarios no expertos, limitando así el aprovechamiento ciudadano de los datos. Mostrando esto, que esta situación no es exclusiva de nuestro territorio. Por otro lado, se suma un punto clave, la brecha evidenciada en el reporte de inclusión financiera del 2024, la cual presenta la información más reciente emitida por la superintendencia financiera de Colombia exponiendo cifras en relación con los créditos activos y sus respectivas limitaciones, dentro de las cuales se ven unas diferencias significativas entre género, grupo etario y ubicación geográfica. Si bien las razones por las cuales pueden llegar a existir desigualdades sobre la inclusión financiera en términos de créditos son amplias, surge la necesidad de cuestionarse si ¿la falta de información de la población adulta en Colombia haga que la brecha de inclusión financiera persista?

3. Justificación

A partir de este trabajo se tienen tres motivaciones que respaldan el porqué es necesario y tiene la importancia de su aplicación, la primera motivación es la falta de herramientas disponibles de este tipo dentro de la perspectiva general del problema evidenciado, al ser una alternativa moderna e impulsada por la inteligencia artificial no se encuentra publicada ninguna plataforma o recurso que cumpla con la finalidad mencionada que se quiere desarrollar dentro de este alcance, sin embargo se destacan estudios en relación a la precepción de los usuarios de los datos abiertos como el de (Segura, 2023) el cual desarrolló una plataforma que analiza la metadatos de los archivos subidos al portal de datos abiertos de Colombia con el fin de encontrar relaciones con la utilidad percibida por parte de los agentes. Como segundo aspecto aportar a reducir la brecha de desinformación en temas financieros con el fin de lograr que las personas puedan tomar decisiones informadas de la mano de datos públicos verídicos es un punto de valor importante, de acuerdo con esto según el estudio de Nyarko, Amoateng y Aboagye (2023) se centra en los efectos positivos que la inclusión financiera y otras características ofrecen para impulsar el progreso a largo plazo, expone como al atacar la inclusión financiera son notables los cambios hacia la reducción de pobreza en economías en vía de desarrollo. Por último, tener en mente una visión proyectada hacia la implementación de este tipo de herramientas en más ámbitos de interés social, que tengan la relevancia e impacto para la población adulta en Colombia, es un estímulo para enfrentar esta problemática.

4. Objetivos

Objetivo General

Desarrollar una herramienta que permita facilitar el uso de los datos abiertos relacionados con tasas de interés activas en Colombia, a partir de la traducción de consultas en lenguaje natural a SQL por medio de un modelo de lenguaje a gran escala.

Objetivos específicos

- Explorar el uso y rendimiento de diferentes LLM mediante métricas de desempeño con el fin de determinar aquel que ofrecerá mejores resultados para aplicarlo al contexto del problema.
- Definir y preparar los datos escogidos del portal de datos abiertos de Colombia con el fin de garantizar la consistencia y calidad de la información dentro del procesamiento.
- Implementar el modelo seleccionado tomando en cuenta los diferentes ajustes y técnicas adicionales necesarias para optimizar el rendimiento en la generación de resultados, así como generar la interfaz adecuada para el uso efectivo de la herramienta.
- Evaluar el desempeño del modelo mediante diferentes métricas que permitan garantizar su funcionamiento, así como el rendimiento y la usabilidad de la interfaz desarrollada.

5. Marco Teórico

5.1 Gobierno de datos abiertos

El gobierno de datos abiertos es un enfoque ejecutivo que se ha llevado a cabo con el respaldo desde el derecho fundamental del acceso a la información pública, manifestado en el artículo 74 de la constitución política de 1991 a lo largo del tiempo. Sin embargo, dado los cambios y evoluciones del sistema de comunicaciones globales entorno a la tecnología a partir de la ley 1712 del 2014 sobre la ley de transparencia y del derecho de acceso a la información pública, se profundizo en todas las oportunidades, libertades y limitaciones sobre el uso y acceso de los datos públicos, que son todo conjunto de datos no procesados que está bajo el control de alguna entidad u órgano del gobierno (Gobierno de Colombia, 2025). Este marco normativo brinda una visibilidad clara entorno a los objetivos principales de la publicación del material dentro de los cuales se resalta la búsqueda de una ciudadanía informada, crítica y con criterio frente al manejo y actuación del gobierno, el empoderamiento de las personas y la democratización de la información. El gobierno a través del ministerio de justicia hace hincapié en el siguiente punto relacionado a la ciudadanía informada y sus efectos positivos alegación que respalda de manera muy afín las motivaciones planteadas:

Uno de los grandes retos es contar con una ciudadanía bien informada acerca del cómo se hace efectivo su derecho de acceso a la información, no sólo porque fomenta la transparencia en la gestión pública, el buen gobierno y la eficiencia administrativa a través de la rendición de cuentas y la exigibilidad de resultados, sino porque una ciudadanía informada puede ejercer plenamente otros derechos fundamentales y tomar decisiones más acertadas. Así, una ciudadanía bien informada, es una ciudadanía que conoce sus derechos,

que los ejerce plenamente, que exige rendición de cuentas y que participa en los asuntos públicos. (Ministerio de justicia y del derecho, s.f, p.3)

Adicionalmente sobresale el artículo 8 de la sección II de publicidad y contenido de la información donde se crea como objetivo poder realizar la gestión para que personas de todas las comunidades, dialectos y culturas diferentes se les garantice un formato adaptado para su uso y entendimiento. (Ministerio de justicia y del derecho, s.f, p.15)

De este punto surge la fuente de información primaria que nos servirá como pilar fundamental para atacar una de las múltiples problemáticas sociales que existen en Colombia y cambiar la visión del uso de los datos abiertos no solo como un mecanismo de transparencia sino como una herramienta que se proyecta en una oportunidad estratégica para la toma de decisiones.

5.2 Inclusión financiera

La inclusión financiera en Colombia es la medición que se le realiza al acceso sobre diferentes productos financieros asequibles formales y que realizan un uso responsable de estos esta información, este indicador es un reflejo sobre el desarrollo y salud financiera del país así como una palanca para enfrentar el problema de la pobreza.(Banco mundial,2024) En Colombia el ente encargado de realizar un seguimiento y actualización de estos indicadores es la Superintendencia Financiera de Colombia que por medio de reportes anuales comparte los hallazgos encontrados sobre el tema con aperturas relevantes, ya sean de productos de depósito, estructura financiera y transaccional, accesos a seguros y créditos, para el presente año se encontraron los siguientes puntos sobre créditos:

1. Los hombres acceden más al crédito que las mujeres, excepto en el sector formal no financiero (acceden más al sector real comercio y a las microfinancieras no vigiladas)

2. La brecha etaria se revierte entre jóvenes por su alto acceso al sector real comercio, mientras que se amplía entre mayores por su baja presencia en este sector.
3. La brecha de acceso al crédito entre ciudades y municipios rurales dispersos se amplió, con aumentos generalizados en zonas urbanas para todos los grupos de entidades.
4. El ranking regional se mantiene, pero el rezago del Pacífico se acentúa, dado que es superado por el Llano debido a un mayor acceso al sector formal no financiero. El Caribe se mantiene de último.

Puntos los cuales resaltan las falencias actuales en cuanto al campo de la inclusión financiera. Esta información reside en los mismos repositorios gubernamentales los cuales podrían ser consultados por los ciudadanos sin la dependencia de un informe anual. Sin embargo, esto abre una oportunidad de evolución hacia un puente de conexión entre la educación financiera y el exponencial crecimiento del aprendizaje de maquina el cual genera una respuesta personalizada a las consultas de las personas.

5.3 Procesamiento de lenguaje natural (PNL)

El uso del lenguaje natural es una estrategia fundamental que hemos usados como humanos por miles de años atrás. Este concepto es como llegamos a definir la comunicación y difusión de conocimiento entre personas. Debido al alza en el uso de diferentes formatos digitales y la creciente exigencia de interactuar con la tecnología, donde hemos compartido información ya sea en redes sociales, páginas web o repositorios con connotaciones enfocadas al lenguaje natural como llamadas, opiniones o videos. El PNL surge como una solución dentro de los LLM, la necesidad de procesar nuevas fuentes y tipos más complejos de información realizando el restablecimiento y extracción de información contemplando las

bases formales del lenguaje y dándoles una representación semejante en un formato objetivo a partir del procesamiento computacional (Ruiz,2021).

Dentro del marco del proyecto actual la labor de traducir la expresión natural en español de una pregunta en texto, a un lenguaje técnico que es SQL, parte de los LLM donde existe una arquitectura muy usada para el desarrollo de modelos a partir de redes neuronales, creada en el 2017 por investigadores que trabajaban en Google, llamada *Transformers* la cual es una estructura que brindó un salto inmenso sobre las capacidades de los modelos, este se centró en el uso de un mecanismo de auto atención que logra evaluar que tan importante es una palabra dentro de una cadena, captando dependencias y conexiones de alto alcance dentro del análisis, al igual que el uso de paralelismo dentro del entrenamiento, potenciando las soluciones dentro del NLP. Hoy en día esta arquitectura se ve presente en los modelos más actuales en su campo como GPT, T5, BERT, Flan-T5, sin embargo, no es la única forma en la que estos son clasificados, bajando un rango dentro de los *transformers* se encuentran los tipos *encoder-only*, *decoder only* y *encoder-decoder*, los cuales se alinean para problemas de diferentes connotaciones, dentro del campo del text to SQL las arquitecturas según la literatura extraída que presentan un buen comportamiento son las de tipo *encoder-decoder* y los de tipo (*Retrieval-Augmented Generation*, RAG) entre otros; los cuales siguen la misma arquitectura con un punto adicional clave y es la búsqueda externa dentro de un módulo de recuperación sobre información especializada del problema, lo que le brinda ventajas tanto en adaptabilidad como mejor interpretación. Este campo de conocimiento dentro del lenguaje natural entra con amplio potencial, donde justamente a través del uso del aprendizaje de máquina y aprendizaje supervisado, hace uso de los datos de entrenamiento para guiar al modelo a reconocer los patrones que satisfagan la estructura del problema. Este proceso se

fundamenta en diversas técnicas de procesamiento orientadas a la representación y comprensión de la información expresada en lenguaje natural. Por otro lado, los modelos pertenecientes a este campo de los LLM tienen la característica de una gran cantidad de parámetros dentro de su red neuronal en un rango dentro de los (0.5B-675B) dependiendo del tamaño de este, por lo que resulta ser una limitación en cuanto al poder de cómputo el uso de los modelos categorizados como robustos en máquinas de uso personal, a pesar de esto se han desarrollado diferentes técnicas que brindan una opción efectiva para que estos obstáculos sean evitados.

Especialmente dentro de área del conocimiento del PNL se encuentra el *text to SQL* el cual nace alrededor de las décadas de 1970 la cual fue adaptándose al progreso y a las soluciones desarrolladas por investigadores, llegando a tener una relevancia significativa en relación con las problemáticas que se quieren enfrentar hoy en día.

5.4 Campo del *Text to SQL*

El proceso *text to SQL* en este proyecto se construye a partir de un *pipeline* que integra tres piezas clave: el modelo a utilizar, la técnica de ajuste eficiente LoRA y el mecanismo de decodificación Parsing *Incrementally for Constrained Auto-Regressive Decoding* (PICARD). En primer lugar, el modelo a utilizar por ser una tipología multilingüe permite comprender preguntas en español y transformarlas en sentencias SQL. Luego, con LoRA se entrena al modelo de manera supervisada para que aprenda el estilo y la estructura de las consultas en el dominio específico, sin necesidad de modificar todos sus parámetros, lo que hace el proceso más rápido y eficaz. Finalmente, durante la generación de la sentencia, PICARD actúa como una especie de segmentación de resultados pues se asegura de que el SQL producido sea sintácticamente correcto. De esta manera, la ruta combina comprensión

diferentes puntos claves que aseguran robustez y validez de los resultados que se generen. El detallado de la implementación de cada uno de los pasos se encuentra en la metodología de la investigación.

5.5 Redes Neuronales

Las redes neuronales artificiales son modelos computacionales inspirados en el funcionamiento del cerebro humano, diseñados para identificar patrones, aprender de los datos y realizar diferentes tipos de tareas. Estas redes están compuestas por unidades básicas llamadas neuronas artificiales, las cuales se organizan en capas y se encuentran interconectadas entre sí mediante pesos.

Una red neuronal típica se estructura en tres tipos de capas: capa de entrada, donde se reciben los datos; capas ocultas, encargadas de procesar la información mediante transformaciones no lineales; y capa de salida, que genera el resultado final del modelo. Como se representa en la siguiente imagen Figura 12. Cada neurona combina las entradas ponderadas, aplica una función de activación y transmite la información a la siguiente capa.

El proceso de aprendizaje de una red neuronal se basa en el ajuste iterativo de los pesos a través de algoritmos de optimización, utilizando un conjunto de datos de entrenamiento. Este ajuste se realiza minimizando una función de error que mide la diferencia entre las predicciones del modelo y los valores reales.

Gracias a su capacidad para modelar relaciones complejas y no lineales, las redes neuronales han sido ampliamente utilizadas en áreas procesamiento de lenguaje natural. Dentro de este contexto, las redes neuronales se emplean como una herramienta fundamental

para el análisis y modelado de datos, permitiendo obtener resultados más precisos y robustos frente a métodos tradicionales.

A partir de la evolución de las redes neuronales profundas, surgen arquitecturas más avanzadas orientadas al manejo eficiente de grandes volúmenes de información y dependencias complejas en los datos. Entre estas arquitecturas destacan los Transformers**,** un tipo de modelo basado en mecanismos de autoatención (*self-attention*) que permite procesar secuencias de datos de manera paralela y capturar relaciones de largo alcance de forma más efectiva que enfoques tradicionales.

5.6 Arquitectura Transformer autoregresivo

Dentro de la siguiente sección se descompondrá por partes los diferentes bloques que hacen parte de la arquitectura de *Transformers,* conceptos los cuales son necesarios para entender el funcionamiento, bases matemáticas y flujo de la información dentro del modelo. Así como se presenta en la tabla 7, la cual contiene los valores de los parámetros principales del modelo Mistral-7B como ejemplo.

5.6.1 Word Embeddings y RoPE

Como bien se sabe en el campo del procesamiento de lenguaje natural el uso de palabras para interactuar con un modelo de inteligencia artificial suena a una tarea imposible, y lo es sin las transformación necesarias, simplemente con el uso de palabras como “ayuda”, “necesito” o “respuesta” las maquinas no serían capaces de producir un resultado de acuerdo a nuestras necesidades, la forma correcta de tener esta conexión es a través de los números, la transformación de palabras a números ha sido una técnica que ha evolucionado con el tiempo, enfocada cada vez más a exponer el significado tanto sintáctico, semántico y

lingüístico propio de la palabra. Por lo que esta tarea ha llegado a representarse por medio de vectores en determinado espacio vectorial n-dimensional denominados *embeddings*. En este sentido (Su Et al.,2021) nos presenta un método alternativo para poder extraer la mayor cantidad de información de los vectores al tomar en cuenta 2 grandes métodos que se estaban usando *relative positional embeddings* y *absolute positional embeddings* para así tomar lo mejor de estos métodos la rotación posicional de vectores (RoPE) hace uso de dentro del mecanismo de atención la relación entre palabras dependa de la distancia entre ellas y no solo de su posición absoluta. Un punto a favor es que el uso de esta técnica no crea parámetros nuevos ni aumento el tamaño del modelo y si introduce información relativa de manera natural. Se presenta la representación matemática de la transformación posicional.

$$q_m = R_m W_q x_m$$

$$k_n = R_n W_k x_n$$

Donde:

- x_m: representación del token en la posición m.
- W_q: matriz de proyección para Query.
- W_k: matriz de proyección para Key.
- R_m y R_n: matrices de rotación dependientes de la posición.

Posteriormente pasan los vectores resultantes al mecanismo de atención el cual será enunciado en el siguiente apartado.

5.6.2 Self attention causal

En modelos de lenguaje autorregresivos como Mistral 7B-v1, el objetivo de entrenamiento consiste en modelar la probabilidad condicional del siguiente token dada la secuencia previa:

$$P(x_1, x_2, \dots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_{<t})$$

Donde $x_{<t}$ representa todos los tokens anteriores a la posición t.

Para garantizar que el modelo no acceda a información futura durante el entrenamiento, se emplea un mecanismo denominado *self-attention causal*. Sea $X \in \mathbb{R}^{T \times d_{model}}$ la matriz de *embeddings* de entrada. Las proyecciones lineales generan:

$$Q = XW_Q, K = XW_K, V = XW_V$$

donde:

- $W_Q, W_K, W_V \in \mathbb{R}^{d_{model} \times d_k}$
- $d_k$es la dimensión por cabeza de atención.

La atención estándar se define como:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

En el caso autorregresivo, se introduce una máscara triangular inferior $M \in \mathbb{R}^{T \times T}$, definida como:

$$M_{ij} = \begin{cases} 0 & \text{si } j \leq i \\ -\infty & \text{si } j > i \end{cases}$$

La atención causal se define entonces como:

$$\text{Attention}_{causal}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

El término M fuerza a que:

$$\alpha_{ij} = 0 \text{ para } j > i$$

Impidiendo que la posición i atienda a posiciones futuras, la máscara causal asegura que cada token x_t únicamente dependa de $\{x_1, x_2, \ldots, x_t\}$ Preservando la propiedad autorregresiva requerida para la generación secuencial. El modelo se entrena minimizando la entropía cruzada:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(x_t \mid x_{<t})$$

Gracias a la máscara causal, el cálculo de $P(x_t \mid x_{<t})$ respeta estrictamente la dirección temporal.

5.6.3 Multi-head attention

La Multi-Head Attention (MHA) es un mecanismo fundamental en la arquitectura Transformer, introducida en el trabajo *Attention Is All You Need*, cuyo propósito es permitir al modelo capturar múltiples tipos de dependencias dentro de una secuencia de manera

simultánea. A diferencia de una atención simple, que genera una única representación contextual, la atención multi-cabeza proyecta la entrada en múltiples subespacios y calcula atención en cada uno de ellos de forma paralela. Sea una secuencia de entrada representada como:

$$X \in \mathbb{R}^{T \times d_{model}}$$

donde:

- Tes la longitud de la secuencia.
- d_{model}es la dimensión del modelo.

La atención multi-cabeza consta de h cabezas independientes. Para cada cabeza i, se definen tres proyecciones lineales aprendibles:

$$Q_i = XW_Q^{(i)}, K_i = XW_K^{(i)}, V_i = XW_V^{(i)}$$

con:

$$W_Q^{(i)}, W_K^{(i)}, W_V^{(i)} \in \mathbb{R}^{d_{model} \times d_k}$$

y típicamente:

$$d_k = \frac{d_{model}}{h}$$

Para cada cabeza i, la atención se calcula como:

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

El término $\sqrt{d_k}$actúa como factor de normalización para estabilizar los gradientes durante el entrenamiento. Las salidas de todas las cabezas se concatenan:

$$H = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)$$

Posteriormente, se aplica una proyección lineal final $\text{MHA}(X) = HW_O$ donde $W_O \in \mathbb{R}^{hd_k \times d_{model}}$. Esta transformación combina la información capturada por cada cabeza en una única representación con dimensión d_{model}.

5.6.4 Grouped query attention

El Grouped Query Attention (GQA) es una variante eficiente del mecanismo de Multi-Head Attention diseñada para reducir el consumo de memoria y el costo computacional durante la inferencia en modelos de lenguaje autorregresivos de gran escala. Su principal objetivo es disminuir la redundancia en las proyecciones de *keys* y *values*, manteniendo múltiples proyecciones independientes de *queries*. En la Multi-Head Attention estándar, cada una de las h cabezas posee sus propias matrices:

$$W_Q^{(i)}, W_K^{(i)}, W_V^{(i)}$$

Lo que implica que durante la generación autorregresiva deben almacenarse h conjuntos distintos de Ky V en el denominado *KV-cache*. Esto incrementa significativamente el uso de memoria cuando la secuencia es larga. GQA propone reducir este costo agrupando las proyecciones de *keys* y *values*. Sea:

$$X \in \mathbb{R}^{T \times d_{model}}$$

Supóngase que el número total de cabezas de consulta es h_q, pero solo existen $h_k$conjuntos distintos de *keys* y *values*, con $h_k < h_q$. Definiendo el factor de agrupamiento como $g = \frac{h_q}{h_k}$ las *queries* se proyectan de manera independiente $Q_i = XW_Q^{(i)}, i = 1, \dots, h_q$

mientras que las *keys* y *values* se proyectan por grupo $K_j = XW_K^{(j)}, V_j = XW_V^{(j)}, j = 1, \dots, h_k$ Cada conjunto (K_j, V_j)es compartido por g cabezas de consulta. Para cada cabeza de consulta i, la atención se calcula como:

$$\text{head}_i = \text{softmax}\left(\frac{Q_i K_{group(i)}^T}{\sqrt{d_k}}\right) V_{group(i)}$$

Donde $group(i)$indica el índice del grupo de *keys* y *values* asociado a la cabeza i.

5.6.5 Sliding window attention

La *Sliding Window Attention* es un mecanismo de atención utilizado en algunos modelos de lenguaje basados en *transformers*, con el objetivo de reducir el costo computacional asociado al cálculo de la atención en secuencias largas. A diferencia del mecanismo de atención completo del *transformer*, donde cada token puede atender a todos los demás tokens de la secuencia, esta restringe la atención a una ventana local de tamaño fijo que se desplaza a lo largo de la secuencia. De esta manera, cada token solo interactúa con un número limitado de tokens cercanos dentro de dicha ventana, lo que permite disminuir significativamente la complejidad computacional y el uso de memoria, pasando de un crecimiento cuadrático respecto a la longitud de la secuencia a un crecimiento aproximadamente lineal con el tamaño de la ventana. Este enfoque permite procesar contextos largos de forma más eficiente, manteniendo información contextual relevante sin incurrir en los altos costos de la atención completa. (Fu et al., 2025)

5.6.6 Feed forward network

El *Feed Forward Network* (FNN) es un componente fundamental dentro de los modelos transformer, como Mistral 7B, y corresponde a una red neuronal completamente

conectada que se aplica de manera independiente a cada token de entrada. Está compuesto típicamente por dos transformaciones lineales separadas por una función de activación no lineal. Su función principal es introducir no linealidad en el modelo, permitiendo transformar las representaciones generadas por el mecanismo de atención en características más complejas y expresivas.(Vaswani et al.,2017) El mecanismo de atención captura relaciones entre diferentes palabras o tokens en una secuencia, el FFN procesa cada uno de estos vectores de forma individual para enriquecer su representación interna.

Dentro de la arquitectura *transformer*, se ubica inmediatamente después del bloque de autoatención en cada capa del modelo. Es decir, cada capa sigue una estructura donde primero se aplica la atención para integrar información contextual, y luego el FFN para transformar esa información de manera más profunda. Este componente es importante porque permite al modelo aprender patrones complejos y realizar tareas como generación de texto, traducción automática o, en nuestro caso, generación de consultas SQL a partir de lenguaje natural. En este sentido, contribuye a mapear las representaciones semánticas obtenidas del texto hacia estructuras más específicas, como operadores SQL, agregaciones o condiciones, complementando así el papel del mecanismo de atención en el procesamiento del lenguaje.

5.7 Técnica supervisada de Fine-tunig LoRA

En el principio base del fine-tunig la idea es ajustar todos los pesos de las capas internas del modelo a utilizar, con el fin de adaptarlo a los nuevos datos tratados, no obstante, debido al costo computacional de transformar esos pesos, se ha desarrollado la técnica de LoRA que busca descomponer la matriz de pesos en 2 matrices de menos dimensiones con

el fin de no ajustar la totalidad de los pesos, garantizando un buen rendimiento tanto en resultados como en procesamiento. En un modelo base se tiene:

$$h = W\,x$$

Donde:

- $W \in \mathbb{R}^{d \times k}$ son los pesos pre entrenados
- x es la entrada
- h es la salida de la transformación lineal

En el modelo LoRA asumimos que la actualización de bajo rango es:

$$\Delta W = BA$$

Donde:

- $A \in \mathbb{R}^{r \times k}$
- $B \in \mathbb{R}^{d \times r}$
- $r \ll \min(d, k)$

El nuevo peso afectivo sería:

$$W' = W + \Delta W = W + BA$$

Por lo tanto, en entrenamiento el cálculo para la salida de la técnica LoRA es:

$$h = Wx + BAx$$

- Wx: es la parte que se mantiene del modelo preentrenado.
- BAx: es la corrección efectuada con menos parámetros de la forma $\big(r\,(d + k)\big)$ en lugar de $d \times k$

El hiperparámetro r es un componente fundamental para el funcionamiento del método LoRA representa el rango de las matrices de actualización A y B, es decir, el número de dimensiones latentes a través de las cuales se modela el cambio en los pesos. Físicamente, r puede interpretarse como la cantidad de "direcciones" o "conceptos" que el modelo necesita

aprender del nuevo dominio: un valor de r pequeño asume que la adaptación requerida es de baja complejidad intrínseca y puede capturarse en un subespacio reducido del espacio original de pesos.

Esta elección impacta directamente en el costo computacional, ya que en lugar de actualizar una matriz de $d \times d = d^2$parámetros, LoRA entrena únicamente dos matrices: $A \in \mathbb{R}^{d \times r}$y $B \in \mathbb{R}^{r \times d}$, con un total de $2 \cdot d \cdot r$ parámetros entrenables. Dado que en la práctica $r \ll d$(donde los valores típicos son $r \in \{4,8,16,32\}$ frente a d que puede ser de cientos o miles), la reducción en parámetros es sustancial.

6. Metodología

De acuerdo con lo expuesto hasta este punto, con la literatura encontrada, la definición del problema a enfrentar y las limitaciones existentes para hallar alguna herramienta tecnológica; que pueda atacar directamente a la educación financiera, se optó por desarrollar una metodología dividida en 5 etapas:

1. **Exploración y *benchmarking*** para diferentes modelos dentro de los LLM de código abierto, el objetivo principal es evaluar a partir de diferentes métricas distintos modelos con un mismo esquema de datos de referencia para escoger aquel que funciona de mejor manera.
2. Se realizará **la preparación y preprocesamiento** de los datos a utilizar, al igual que la creación de la base de pares (texto y SQL) para utilizarlos en el *fine-tunning* del modelo.

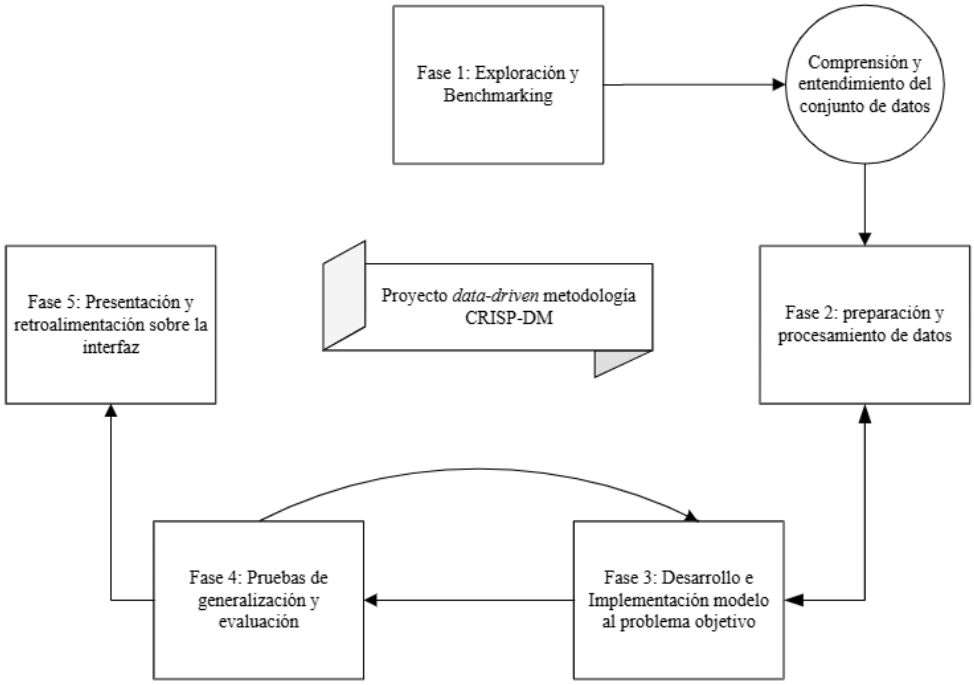
3. Con el resultado de la primera fase se obtendrá el punto de partida para la **implementación al esquema del problema objetivo** realizando la respectiva adaptación y configuración de hiperparámetros del modelo.
4. **Realizar las pruebas de generalizació**n del modelo refinado, evaluando su desempeño general.
5. Por último, la **Presentación de la interfaz** al usuario final en un formato que garantice la usabilidad de la herramienta, de igual importancia recopilar una retroalimentación con el fin de una mejora en estudios futuros.

Como paso siguiente se detallarán con más profundidad cada una de las etapas para mayor claridad, de este modo se garantizará un desarrollo completo y sistemático de la herramienta, tomando en cuenta tanto los requerimientos técnicos como no funcionales del resultado al que se quiere llegar.

Para tener una mejor comprensión del flujo que se va a desarrollar, se presentara un diagrama con las fases y sus interacciones.

Figura 1

Flujo de trabajo de la metodología propuesta basada en CRISP-DM



Nota. Elaboración propia (2025).

Este flujo de trabajo se basó en la metodología CRISP-DM la cual tiene un enfoque iterativo que permite que conforme se adquiere más conocimiento por medio de las iteraciones se perfeccione el resultado esperado, lo cual es especialmente afín a un proyecto con una óptica hacia los datos.

Para esta primera fase se proponen las siguientes actividades:

1. **Investigación previa**: Recopilar por medio de las investigaciones existente los modelos actuales más utilizados en el campo del *text to SQL (transformers)*, tomando en cuenta sus características y diferentes ajustes de optimización.

2. **Selección de modelos**: Escoger los modelos con mejor prospección a partir de la investigación realizada, considerando su aplicabilidad al problema planteado con las limitaciones presentes.
3. **Comparación de desempeños**: Evaluar el desempeño de los modelos seleccionados utilizando diferentes métricas y un conjunto de datos estándar para garantizar una comparación imparcial.
4. **Selección final**: Identificar el modelo que presente el mejor desempeño en general y mayor potencial de implementación en el contexto delimitado.

Se evidenciarán los resultados obtenidos por medio de tablas comparativas, con datos adicionales y observaciones sobre la implementación de los modelos escogidos, estos recursos serán documentados y almacenados, al igual que los notebooks usados, este registro se realizará tanto en el documento de la investigación como en un repositorio de GitHub para tener una trazabilidad.

Como segunda fase se encuentra la definición, preparación, procesamiento de los datos elegidos del portal de datos abiertos de Colombia en el área de hacienda y crédito público, el cual tienen alrededor de un millón de registros los cuales cronológicamente parten de octubre del 2013, este conjunto de datos contiene los registros de:

2

Las tasas de interés promedio ponderadas, el monto y el número de créditos desembolsados durante la semana para los diferentes tipos de crédito (consumo, vivienda, comercial ordinario, comercial preferencial o corporativo, comercial tesorería, comercial especial y créditos productivos), clasificadas por tipo de persona, sexo, tamaño de empresa, tipo de crédito, tipo de garantía, producto de crédito, plazo del crédito, grupo étnico, antigüedad de la empresa, tipo de tasa, rango por monto desembolsado en salarios

mínimos legales mensuales vigentes (SMLMV), clase de deudor, código CIIU y código del municipio, de los establecimientos de crédito.(Superintendencia financiera de Colombia,2013)

Como principales actividades se encuentran:

1. **Limpieza**: Identificar inconsistencias o errores en el conjunto de datos para su posterior cambio con la finalidad de garantizar su calidad y fiabilidad.
2. **Depuración y filtrado de datos**: Aplicar criterios de selección y filtrado para conservar únicamente los campos que permitan consultas de gran valor al igual que eliminar registros redundantes o no significativos.
3. **Creación de conjunto de datos pares**: Generar un conjunto de datos estructurado en pares de entrada y salida que sirva como base para el entrenamiento y evaluación del modelo.

De la creación del insumo de la tercera actividad se reflejará un impacto sobre el entrenamiento y su calidad en el paso del *fine-tunig*, debido a que con este se le refinara al modelo el esquema sobre el cual tiene que generar consultas, ya sea sobre las columnas y sus diferentes valores posibles. El tamaño y la variedad de esta base determinara una gran parte de los resultados generados por lo cual se buscará tener alrededor de 300 registros y en caso de que mediante el uso de este se vea necesario aumentar, se optara tanto por el aumento sintético de los registros; por medio de técnicas que buscan sinónimos sobre las preguntas y crean variedad lingüística, o la creación natural de los registros, Como resultado de esta fase se tendrá la certeza de la fiabilidad y condiciones óptimas de los datos a utilizar así como la base para iniciar el entrenamiento del modelo en pasos siguientes, como entregables estarán los *scripts* y *dataframe* generado que serán expuestos en el repositorio de GitHub creado.

Dentro de la fase 3 y 4 se encuentra la implementación del modelo final seleccionado y la evaluación de este, este proceso representara un paso crítico dentro del desarrollo del proyecto pues de este dependen tanto los resultados como la creación de la herramienta.

De las actividades involucradas se destacan:

1. **Entrenamiento del modelo**: Entrenar el modelo elegido con el conjunto de datos que se desarrolló en la fase previa, asegurando consistencia y calidad en las consultas ingestadas al modelo.
2. **Ajustes y optimizaciones**: Realizar el ajuste continuo de hiperparámetros para mejorar el rendimiento general del modelo.
3. **Validación del modelo**: Evaluar el modelo final utilizando un conjunto de datos de prueba.
4. **Análisis de métricas**: Analizar los resultados obtenidos con métricas propias del campo de *Text-to-SQL*.
5. **Cambios en errores evidenciados**: Identificar puntos débiles dentro de las consultas incorrectas con el fin de realizar cambios y generar mejores resultados.

Estas 2 secciones serán documentadas dentro del repositorio, así también se almacenarán los pesos del modelo final y los ajustes aplicados con el fin de asegurar su reproducibilidad futura. Como resultado general se tendrá un modelo funcional y preparado para su uso.

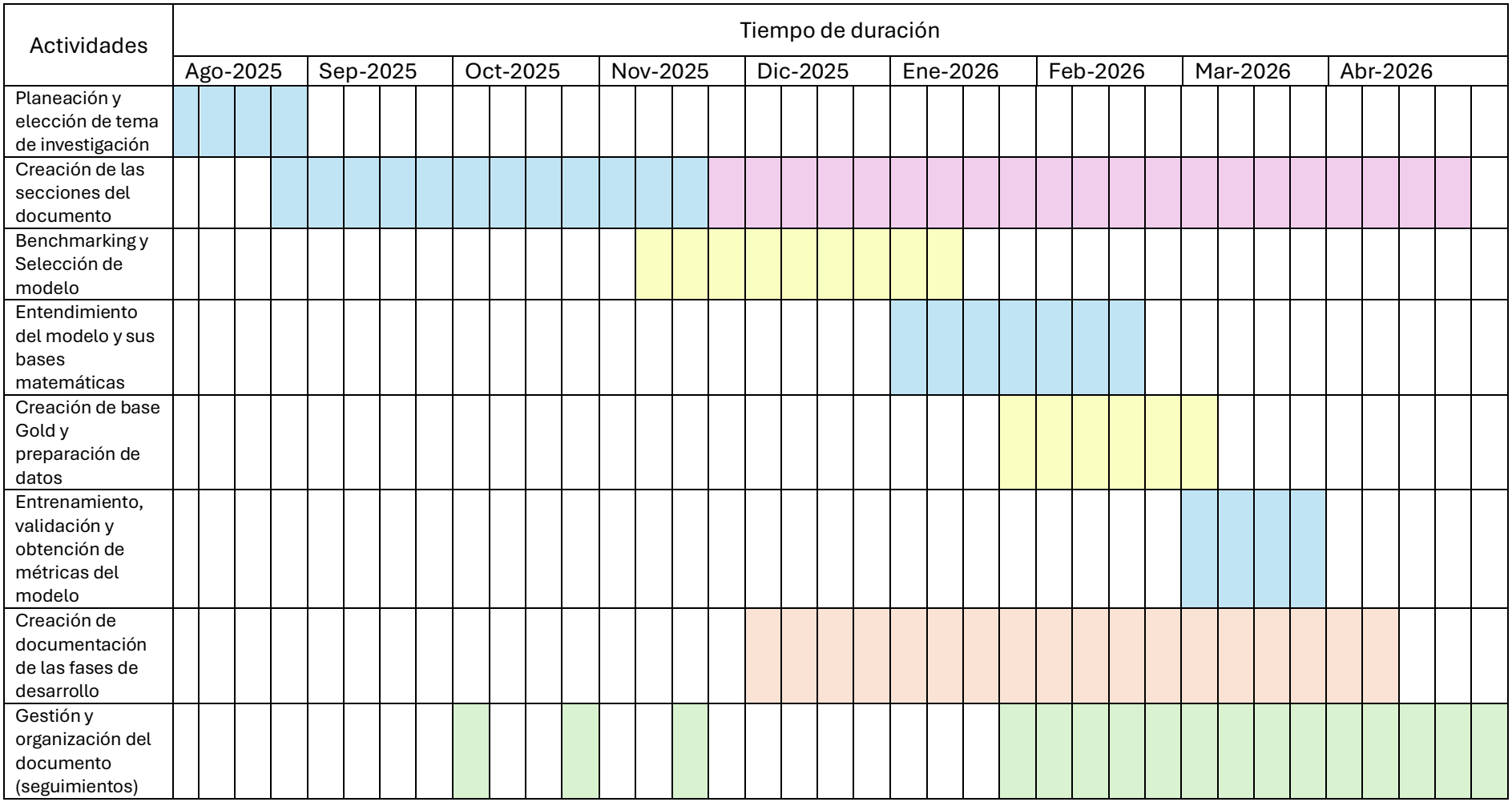
Por último el diseño de la herramienta se va a centrar en garantizar como bien se ha dicho la usabilidad, por lo que no se centrara en diseños elaborados u complejos que puedan obstaculizar la adaptación o enlace del desarrollo principal, en ese sentido se utilizará ya sea una interfaz de consola o notebook la cual soporte la entrada de la pregunta del usuario y

presentara visualmente la respuesta por el mismo medio, a pesar del formato escogido se recopilará una encuesta posterior a su uso sobre algunos usuarios la cual dará un indicio sobre las evoluciones que pueda llegar a tener el proyecto y el rumbo por el cual se debe adaptar la herramienta.

Con la finalidad de tener una ruta cronológicamente clara sobre el paso a paso que se desea realizar en cada una de las etapas de la metodología mencionada, se presentara un diagrama de tipo Gantt el cual evidenciara el plan completo, así mismo se buscara a través de este tener una noción precisa sobre las tareas necesarias para tener un avance continuo y poder gestionar las responsabilidades, se realiza la claridad que esta herramienta no es camisa de fuerza para el desarrollo de las fases se utiliza como guía. De las actividades enunciadas en la siguiente visualización se realiza la claridad dado que cada espacio se representa de manera semanal desde el 1 de agosto del 2025 al 30 de abril del 2026, así mismo los colores usados representan respectivamente:

	Hace referencia a una actividad enfocada en la investigación.
	Es una actividad enfocada a la redacción y alimentación del documento.
	Actividad de desarrollo, exploración tecnológica y uso de habilidades técnicas.
	Creación de documentación técnica diferente al documento escrito.
	Seguimientos con tutores y retroalimentaciones.

7. Cronograma



8. Desarrollo de la metodología

Dentro del siguiente apartado se encontrarán todas las acciones realizadas y guiadas por el marco de la metodología, durante el desarrollo de este proyecto, las cuales fueron de suma importancia para el cumplimiento de los objetivos planteados.

8.1 Exploración inicial y *benchmarking*

Como sección inicial se tendrán en cuenta las aproximaciones que se realizaron durante cada bloque de la etapa uno de la metodología, la cual consta de diferentes hallazgos que condujeron a tomar decisiones para la elección de los modelos a utilizar (*benchmarking*) y al mismo tiempo presentar indicios de las herramientas con potencial uso.

8.1.1 Exploración preliminar

Dentro de las pruebas, recursos, evidencias e información recopilada se destaca: primero la exploración de diferentes tipos de modelos, metodologías y enfoques que existen en el campo del procesamiento del lenguaje natural a través del seguimiento de la literatura actual, este recurso se ha convertido en una marca distintiva en el desarrollo de esta investigación, ya que ha alimentado la visión general y la dirección de los recursos a utilizar, del mismo modo; esto crea oportunidades de realizar mejores elecciones frente a las temáticas de la selección de métricas y afines, así como el reconocimiento de diferentes herramientas y variedad de maneras de implementar modelos. De manera complementaria se realizó interacción con un *notebook* (Roucher,2025) el cual hace uso de 2 diferentes modelos de procesamiento de lenguaje natural llamados: Llama 3.1-8B-Instruct y Qwen3-Next-80B-A3B-Thinking a través de un agente, dichos modelos tienen características similares y se

enfocan en resolver problemas de generación de texto y procesamiento de lenguaje natural, son de fácil acceso debido a la plataforma de Hugging face que cumple el rol de repositorios de modelos de IA y puente entre la interacción de los modelos sin entrenamiento y los usuarios del modelo. Para la implementación de un caso de uso simple se utilizó la tabla 8 que está en el apéndice A.

Para esta representación se realizó la siguiente pregunta en inglés: *¿Can you give me the name of the client who got the most expensive tip?* La figura 13, 13.1 y 13.2 muestran una trazabilidad de 8 pasos en los que el agente genera respuestas y las evalúa en relación con la coherencia, validez de la sentencia y si la ejecución de la consulta no es vacía.

Al finalizar la ejecución muestra en el último paso el resultado efectivo para la respuesta según la tabla de *Receipts* haciendo uso del primer modelo. Asimismo, se realizó estas mismas acciones al segundo agente, presentando el resultado presente en la figura 14 dentro del apéndice B.

A partir de estos casos de uso utilizados, se exploró la viabilidad de su entrenamiento y diferentes pruebas comparando sus métricas y diferentes características frente al procesamiento, así como la implementación de preguntas en español, escalamiento sobre la base de datos que consulta y el comportamiento de los resultados. En consecuencia, se optó por no seguir con esta línea de trabajo y realizar una reorientación del desarrollo debido a limitaciones en el uso de los dos modelos mencionados anteriormente, en específico, la cantidad de ejecuciones posibles se veía ceñida por los recursos computacionales compartidos y solicitudes de inferencia que se tenían disponibles dentro de la plataforma de Hugging Face, lugar donde se alojan los modelos utilizados, lo que hacía que no fuera viable experimentar lo suficiente para lograr cumplir con los objetivos planteados.

Adicionalmente, se desarrolló a partir de la librería de *streamlit* una interfaz la cual hace uso de un LLM de forma cuantizada y muestra de forma amigable al usuario una respuesta intermedia del proceso al que se quiere llegar siendo la consulta de SQL como se evidencia en la figura 15 del apéndice. Se tendrá en cuenta esta opción ya que de forma satisfactoria la implementación se alinea con la finalidad de la investigación. La cual se puede ver en la figura 6 en el Apéndice C

8.1.2 Exploración inicial y elección de modelos

Dentro de la realización de esta primera fase y tomando en cuenta la investigación y rutas tomadas previamente se recopilo información de modelos de lenguaje a gran escala que fueran de código abierto que tuvieran buenos valores en rendimientos y métricas dentro del caso de uso de texto a SQL. Así también la elección de modelos con una cantidad de parámetros acorde a la cantidad de recursos computacionales disponibles sin comprometer el desempeño de las necesidades propuestas.

De acuerdo con estos requerimientos se adaptó el uso de 2 modelos diferentes mistral-7b-v0.1 y SQLcoder-7b-2 de los cuales el primero es un modelo que fue entrenado con un propósito general y diseñado para tareas como comprensión y generación de texto, sus características positivas en razonamiento estructurado y generación de código lo convierten en una alternativa adecuada para la adaptación y uso en el contexto del texto a SQL. Por otro lado, el segundo es un modelo especializado en traducción de lenguaje natural a SQL, por lo que en su preentrenamiento se utilizaron tanto sentencias como preguntas orientadas a diferentes dominios de bases de datos lo que hace que su precisión sea alta y volatilidad en la estructura de la sentencia generada sea baja. En este orden se buscará comparar el rendimiento del modelo especializado con la adaptación del modelo general a una base

especializada del problema de generación de consultas SQL. Por consiguiente, se realizó la elección de los modelos con la finalidad de evaluar la capacidad de adaptación mediante el uso del fine-tuning frente a sus capacidades de un modelo generalista y especializado, evaluando si la versatilidad en el entendimiento del lenguaje natural de mistral superaba las capacidades sintácticas de SQLcoder.

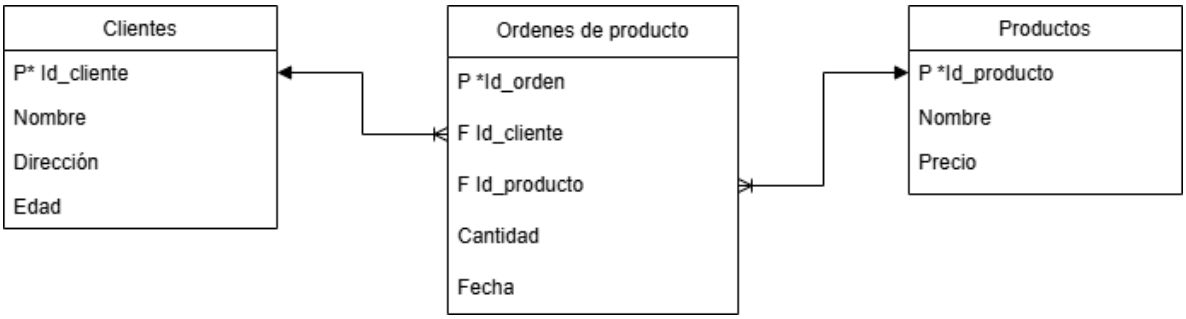
Se destaca el hecho de que la utilización de ambos modelos se realizara de forma cuantizada en un ambiente local debido a las limitaciones computacionales que se tienen en la utilización local de los modelos. Los experimentos iniciales como evaluación de modelos en local para la base de ejemplo y el desarrollo de la interfaz fueron ejecutados en un equipo portátil con procesador Intel® Core™ i5-1035G1 de cuatro núcleos y frecuencia base de 1.00 GHz, acompañado de 8 GB de memoria RAM, de los cuales 7.75 GB se encontraban disponibles para el sistema. El equipo contaba con una unidad de almacenamiento de aproximadamente 954 GB, con una capacidad utilizada de 333 GB al momento de la experimentación. En cuanto al procesamiento gráfico, se utilizó una Intel® UHD Graphics integrada, con 128 MB de memoria gráfica dedicada, por lo que no se dispuso de una GPU especializada para entrenamiento de modelos de aprendizaje profundo la cual fue reemplazada por el uso de Colab configurado con un entorno de ejecución T4. Adicionalmente, se empleó un sistema operativo de 64 bits basado en arquitectura x64, sobre el cual se ejecutaron las herramientas y bibliotecas requeridas para el desarrollo del proyecto.

Evaluando su comportamiento y resultados sobre un esquema de datos preliminar previo a la real, Las pruebas preliminares fueron realizadas utilizando una base de datos de ventas debido a que este tipo de conjuntos de datos son ampliamente utilizados en tareas de generación de consultas SQL y presentan una estructura relacional suficientemente

representativa para validar el funcionamiento inicial del modelo. El propósito de estas pruebas era evaluar aspectos relacionados con la generación de consultas, la conexión con la base de datos, la ejecución de las sentencias SQL y la capacidad general del modelo para traducir preguntas en lenguaje natural a consultas estructuradas. El cual se diferencia tanto en su complejidad estructural y el tamaño de la información contenida. El modelo relacional para el contexto de exploración inicial fue el siguiente, este fue alimentado de tuplas creadas con datos simulados y creados únicamente con la finalidad académica.

Figura 2

Modelo relacional para la base de ejemplo inicial



Nota. Elaboración propia.

Como parte de la preparación de las pruebas se creo una base *Gold* para el contexto experimental el cual consto de 40 registros, recurso el cual se utilizara para evaluar el comportamiento de los 2 modelos con una estrategia de *Few-shot learning* concepto el cual hace referencia a ejecutar inferencias a ambos modelos sin previo conocimiento adicional a la estructura de la base de datos. En este sentido, se realizó la evaluación de ambos modelos con el uso de las 3 métricas: *exact match*, *execution accuracy* y *error rate* las cuales se presentaron e impulsaron su uso dentro de la publicación de en uno de los *datasets* más usados, llamado *Spider,* usado para *benchmarks* entre modelos dentro del campo del texto a

SQL. Su principal diferenciación al resto es la inclusión de distintas bases de datos con múltiples tablas en diferentes dominios, así como sus respectivas sentencias SQL (Yu et al., 2018). Sin embargo, estas tres métricas se alimentan de los siguientes conceptos:

1. **Pregunta**: Requerimiento realizado en lenguaje natural al modelo sobre la base de datos.
2. **SQL *Gold***: Sentencia SQL que representa una respuesta correcta sobre la pregunta realizada (esta se encuentra registrada en la base *Gold* con su respectiva pregunta).
3. **SQL modelo**: Sentencia SQL que genera el modelo como resultado de la inferencia.
4. **Ejecución SQL**: Resultado de la sentencia SQL ejecutada sobre la base de datos.

A continuación, se profundizarán en cada una de ellas frente a su fórmula, interpretación y objetivo dentro de la evaluación del modelo.

Exact match: mide si la consulta SQL del modelo es exactamente igual al SQL *Gold*, esta medida se usa en forma de proporción respecto al total de las predicciones. Tiene la función de expresar la capacidad del modelo de generar consultas óptimas consideradas por el investigador. De igual manera se detalla que la sentencia generada por el modelo y la que esta en la base *gold* deben ser sintácticamente idénticas para que sean contabilizadas dentro de las generaciones correctas. Siendo una métrica ácida, pero relevante dentro de los análisis.

$$EM = \frac{1}{N}\sum_{i=1}^{N} 1(SQL_i^{pred} = SQL_i^{gold})$$

Execution accuracy: mide si la consulta generada por el modelo al ser ejecutada presenta el mismo resultado que la ejecución de la sentencia SQL *Gold*, asimismo se presenta como el número de coincidencias sobre el total de inferencias. Esta métrica tiene la finalidad

de comprobar la efectividad frente a lo requerido, sin tomar en cuenta la sentencia generada, solo el resultado esperado.

$$EX = \frac{1}{N}\sum_{i=1}^{N} 1(Exec(SQL_i^{pred}) = Exec(SQL_i^{gold}))$$

Error rate: toma en cuenta las sentencias generadas por el modelo y verifica cuales de esas no fueron compiladas al ser ejecutadas, ya sea por problemas en la sintaxis, nombre de tablas, o clausulas mal usadas.

$$ER = \frac{1}{N}\sum_{i=1}^{N} 1(SQL_i^{pred}\ produce\ error)$$

Una vez estableciendo dichas métricas, se decidido que: al ser de las primeras pruebas sobre una base de ejemplo con los modelos seleccionados. Se debería detallar los resultados al adicionar un análisis que permita encontrar errores por sintaxis, por esta razón; sobre una serie de preguntas extra las cuales presentan distintos niveles de dificultad, se registraran las consultas generadas explícitamente por los modelos consultar tabla 9 y 10 de la sección apéndice. Se presentan los resultados encontrados para el comportamiento de los dos modelos utilizados sobre la base de ejemplo.

Tabla 1

Tabla de resultados para los modelos evaluados sobre la base ejemplo.

Modelo	Exact match	Execution accuracy	Error rate
Mistral	0%	30%	10%
SQLcoder	0%	40%	7.5%

Nota. Elaboración Propia. Las pruebas se realizaron bajo una base de 40 registros.

De lo encontrado en la tabla anterior podemos destacar como para ambos modelos la métrica de *exact match* no tuvo ninguna coincidencia debido a la falta de información presente a la hora de realizar la inferencia sobre la pregunta, en consecuencia, el modelo no brindo la misma respuesta que la seleccionada como correcta en la base *Gold*. Un punto interesante para resaltar es sobre el *execution accuracy,* que estuvo alrededor del 35% de efectividad, lo cual muestra una capacidad de los modelos por generalizar el contexto y poder generar una respuesta correcta en cuanto a su ejecución. Por último, para el *error rate* se interpreta por medio de las diferencias en las características intrínsecas de ambos modelos pues el SQLcoder está diseñado específicamente para esta tarea por lo que en cuanto a las sentencias generadas tiene una capacidad más amplia de producir menos errores de ejecución mientras que el modelo mistral al no tener una estructura definida puede que use diferentes recursos sintácticos que puede que no pertenezcan al lenguaje SQL.

A partir del análisis de las tablas cinco y seis del Apéndice B se realiza un ejercicio comparativo entre los modelos SQLcoder y Mistral, se observa que ambos presentan un desempeño adecuado en la generación de consultas SQL básicas, particularmente en lo relacionado con el uso correcto de la cláusula SELECT–FROM. En todas las preguntas evaluadas, ambos modelos lograron estructurar correctamente dicha cláusula, lo que evidencia una comprensión sólida de la sintaxis fundamental del lenguaje SQL.

Sin embargo, al analizar consultas de mayor complejidad, se evidencian diferencias relevantes entre los modelos. En las preguntas P3 y P4, que involucran el uso de agregaciones y una estructura más elaborada de la consulta, SQLcoder presentó errores en la generación del SQL correcto, específicamente en el uso adecuado de funciones de agregación. En

comparación Mistral logró resolver correctamente estas consultas, manteniendo coherencia tanto en la estructura general como en el uso de agregaciones.

En cuanto al uso de las sentencias JOIN y GROUP BY, ambos modelos demostraron un desempeño consistente y correcto en los casos donde estas cláusulas eran requeridas, lo que sugiere que los errores observados en SQLCoder no están asociados a la combinación de tablas ni a la agrupación de datos, sino específicamente al manejo de agregaciones dentro de la consulta.

En términos generales, los resultados preliminares indican que Mistral presenta un desempeño más robusto y estable frente a consultas SQL de mayor complejidad, mientras que SQLcoder muestra un rendimiento adecuado en consultas simples, pero presenta limitaciones al incorporar operaciones de agregación.

Por otro lado, se tomó en cuenta el proceso de refinamiento y optimización *prompt engineering* como una variable experimental sobre la base del problema objetivo, el cual hace referencia a la importancia de una buena entrada o petición al modelo sobre su rendimiento, es decir; en muchas situaciones el cambiar una petición sencilla y superficial por una especializada, concreta y objetiva hace que el modelo mejore las métricas por las que está siendo evaluado. Por esa razón se decidió probar a ambos modelos bajo tres diferentes tipos de *prompts*, con la siguiente estructura, una petición o acción a realizar, unas reglas que debería cumplir satisfactoriamente y un esquema de bases de datos asociado. La diferencia recaerá en la instrucción solicitada, donde la primera se redactará de una forma sencilla y genérica mientras que la segunda aparte de ser más detallada en cuanto a la tarea, da un contexto y diferentes reglas de la base de datos manteniendo la superficialidad, la última versión es la más precisa en cuanto a casos de la base de datos, instrucción y contexto.

Todas son similares en cuanto a que dentro de las tres se relaciona el elemento relacional de las tablas existentes de la base de datos. Las versiones precisas y detalladas de los *prompts* se encuentran en la sección C del apartado apéndice. Como principales resultados podemos evidenciar como el modelo de SQLcoder tuvo una tendencia positiva en la métrica de *execution accuracy* mejorando de un 16% a un 30% a medida que el *prompt* es más integral y completo con su función mientras que el modelo de Mistral no tiene una variación notable la cual resaltar frente a esta evaluación. Se realiza la acotación de que los resultados de este proceso solamente reflejan indicios de una posible conclusión y no una realidad rotunda apoyada de pruebas estadísticas que garanticen la significancia. La finalidad de este registro de pruebas será contrastar los resultados obtenidos con las conclusiones evidenciadas en investigaciones realizadas (Shin et al., 2023) debido a que durante el surgimiento de la pregunta ¿Puede un modelo base aplicando técnicas de *prompt engineering* superar el comportamiento de un modelo ajustado por medio de *fine-tuning*? Se han realizado estudios que elucidan una clara posición.

Tabla 2

Resultados de prompt engineering sobre 30 ejemplos.

Versión	Modelo	Exact match	Execution accuracy	Error rate
V1	SQLcoder	0%	16.66%	16.66%
V1	Mistral	6.66%	30%	36.66%
V2	SQLcoder	0%	16.66%	16.66%
V2	Mistral	3.33%	26.66%	33.33%
V3	SQLcoder	0%	30%	10%
V3	Mistral	3.33%	23.33%	20%

Nota. Elaboración propia.

8.2 Preparación y procesamiento de datos

La preparación y procesamiento de datos de este proyecto consta de tres diferentes enfoques. La primera, todos los posibles cambios y transformaciones que se pueden realizar sobre la base del portal de datos abiertos, como normalizaciones de texto, eliminación de registros incompletos y estandarización de formatos. Como segundo, la estructuración de la base de datos relacional a partir de una tabla desnormalizada, con el fin de garantizar la integridad de los datos, consistencia y reducción de redundancias, así como la creación e inclusión de tablas nuevas que complementan y agregan valor a la base de datos. Por último, la creación de la base *Gold* la cual será diseñada en un formato *Jsonl* donde cada registro tiene los campos de instrucción, entrada y salida los cuales cumplen la función de darle la tarea a realizar por el modelo, la pregunta en lenguaje natural y el *target* o la respuesta esperada respectivamente, este recurso representara el eje fundamental sobre el cual realizar la especialización y ajuste por medio del *fine-tunning*.

1. Preparación y transformación, técnicas de limpieza textual:

Hasta la fecha, la base histórica de tasas de interés activas por tipo de crédito que reposa en la página oficial del portal de datos abiertos de Colombia recibió su última actualización el día 8 de abril del 2026 por lo cual cuenta con 74’747.008 registros de créditos. Por esta razón aparte de realizar las respectivas técnicas de normalización de texto y estandarización de formatos, se deberá aplicar un filtro sobre algunas columnas para extraer una cantidad considerable de registros que no limite la extracción valiosa de información, pero sin comprometer las capacidades computacionales operables posibles. Por esa razón se optó por

tomar los siguientes valores para las columnas de fecha de corte, número de créditos desembolsados y plazo de crédito. Las fechas que se tomaron en cuenta fueron los 2 años pasados cargados desde la actualización más reciente es decir el 23 de abril del 2024, para el numero de créditos desembolsados se usó los registros con el valor de uno lo que significa que son registros de un solo préstamo y no la agrupación de muchos créditos, lo que crearía un inconveniente en la interpretación de las tasas promedios generadas, por último el rango de plazo para los créditos se acoto a un rango de 30 días a 6 años el cual representan los tiempos más usuales dentro de los tipos de préstamos demandados en Colombia .Una vez realizado los filtros decantamos a 1’317.647 registro lo cual es una cantidad aceptable y óptima para subir a una base de datos en SQlite. Seguido de este proceso fue necesario realizar un análisis descriptivo que mostrara la organización de los datos e información evidente que permitiera reconocer nuevas tareas de limpieza, como la transformación de las columnas tipo texto a un formato minúscula, la estandarización de los nombres de columnas y la eliminación de todos caracteres que pudieran generar problema a la hora de ejecutar sentencias SQL. Por último, realizar un manejo adecuado de los campos y registros vacíos, en este caso los registros que no tuvieran ningún valor en más de 6 columnas fueron eliminados debido a que se considera que la supresión de estos datos dificulta el entendimiento completo del registro quitando de él la relevancia requerida. Este proceso se realizó por medio de un *notebook* de Colab el cual tiene dividido cada una de las secciones mencionadas.

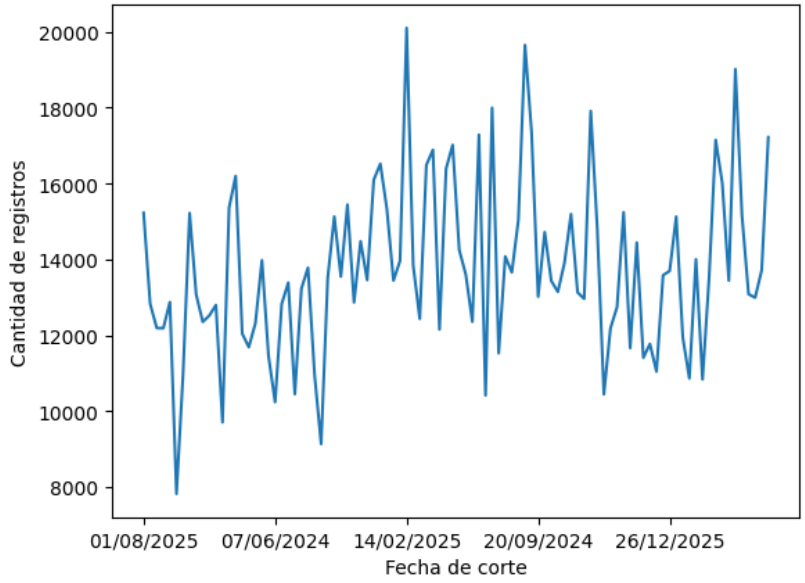
2. Análisis descriptivo sobre la base de datos del problema:

El análisis descriptivo constituye una etapa fundamental en el proceso de comprensión y preparación de los datos, ya que permite identificar las características generales de la

información, al evaluar su calidad y detectar patrones relevantes antes de la creación de la estructura de base de datos. Dado que la mayoría de las variables presentes en la base de datos corresponden a información categórica, el análisis descriptivo se enfocó principalmente en el estudio de frecuencias. Este enfoque permitió identificar la cantidad de categorías existentes en cada variable, así como las categorías predominantes y su frecuencia de aparición dentro del conjunto de datos.

Figura 3

Comportamiento de la variable fecha corte



La variable fecha de corte permite identificar la periodicidad y cobertura temporal de la información utilizada en el estudio. A partir de su análisis se observa que la base de datos contiene registros correspondientes a múltiples períodos de reporte, lo que proporciona una visión histórica del comportamiento de las operaciones de crédito reportadas por las entidades financieras. La distribución de registros por fecha de corte permite verificar la

consistencia temporal de la información y detectar posibles períodos con mayor o menor volumen de observaciones.

Tabla 3

Resultados de análisis descriptivo de variables categóricas relevantes.

Variable	Valores únicos	Moda	Frecuencia
Sexo	5	No aplica	540.291
Tamaño de empresa	5	Microempresa	726.476
Tipo de crédito	5	Crédito productivo	615.866
Plazo de crédito	4	Más de 30 días y hasta 1 año	1’206.059
Producto de crédito	36	Crédito popular productivo urbano	435.564
Tipo de tasa	12	FS	852.413
Nombre entidad	48	Bancolombia	1’268.600

En conjunto, los resultados evidencian una base de datos con alta heterogeneidad en variables relacionadas con productos financieros, entidades y tipos de tasa, mientras que variables como plazo de crédito y tamaño de empresa muestran una concentración más marcada en ciertas categorías. Estas características son relevantes para la etapa de modelado, ya que permitieron identificar desbalances entre categorías y orientar adecuadamente los procesos de transformación y codificación de variables.

3. Estructuración de la base de datos:

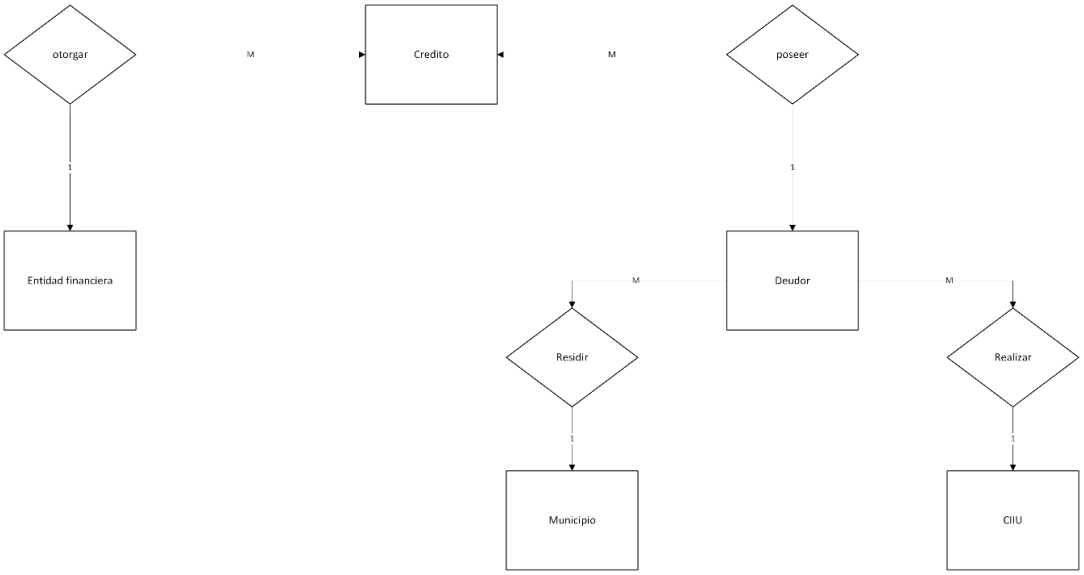
Dentro de la estructuración de la base de datos se debe tener en cuenta que 2 entidades las cuales son Municipio y CIIU no tenían registrada la información asociada a estas entidades propiamente en la fuente de información, de estas se tenían únicamente códigos, de los cuales se revisaron en bases documentales del DANE (departamento administrativo nacional de estadística) y otros, con el fin de cruzarlos con datos adicionales y traer el municipio, departamento asociado al código y la actividad económica. De esta forma se agregaron manualmente cada tabla en formato de Excel al *notebook* realizado.

De acuerdo con las buenas prácticas e instrucciones que recomiendan los autores del libro *Database System Concept* para el diseño de un esquema de bases de datos se sugiere la creación de un modelo de entidad-relación el cual cumple la función de facilitar el entendimiento de las especificaciones necesarias para el contexto. Este recurso representa la estructura lógica final de la base de datos (Silberschatz, Korth & Sudarshan, 2005).

De acuerdo con las estipulaciones necesarias identificadas dentro de la literatura se presenta el modelo entidad relación para el contexto pertinente.

Figura 4

Modelo conceptual entidad relación

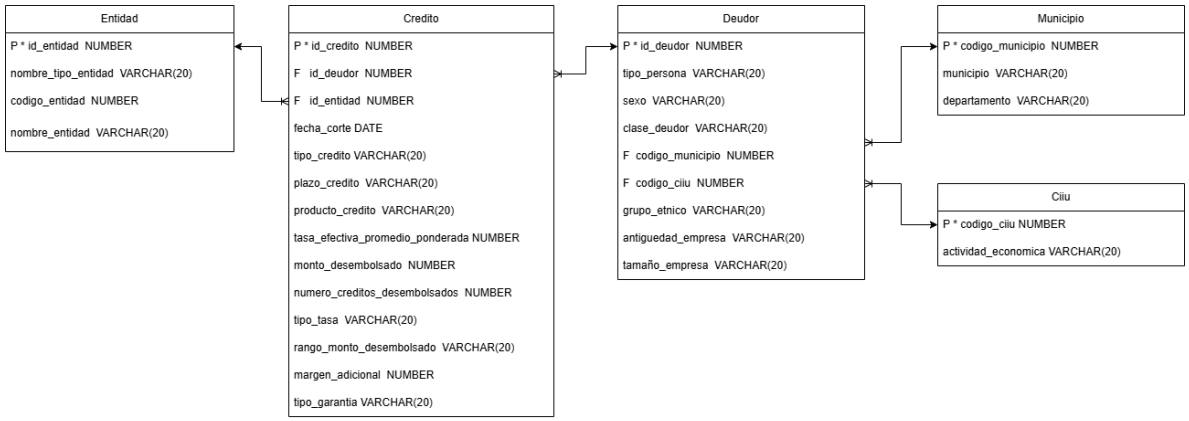


Nota. Elaboración propia

A partir de este diseño nace un esquema más afín y diciente para interpretar la creación de una base de datos tomando en cuenta los atributos dentro de las entidades principales.

Figura 5

Modelo relacional



Nota. Elaboración Propia

Esta estructuración fue posible realizarla debido a la identificación de la naturaleza de los datos en la tabla desnormalizada, la cual se podía agrupar por las diferentes entidades presentadas en el modelo. A partir de las dimensiones de entidad y deudor se eliminaron los registros duplicados con el fin de crear llaves para registros únicos y así poder realizar un

cruce con la información perteneciente a la unidad de crédito. De esta forma se mantiene la consistencia al cruzar la información.

4. Diseño de base *Gold*:

La creación de la base Gold fue un proceso en el cual se tuvo que definir una estructura determinada la cual fuera aceptada por ambos modelos en el momento de realizar el entrenamiento, se utilizó un formato Jsonl en el cual cada línea del archivo corresponde a un registro, para cada registro se necesitaron llenar tres campos *instruction, input y output* de los cuales la primera sección hace referencia a la tarea requerida para cumplir con el objetivo propuesto, la cual para cada uno de los registros de la bases será: “Genera la consulta SQL correcta para la siguiente pregunta.” , la segunda parte es la pregunta en lenguaje natural que se le está enviando al modelo y la última corresponde a una sentencia posible que traduce la pregunta en el lenguaje de SQL. Esta estructura permite generar el entrenamiento necesario para que el modelo afinado no requiera la adición del esquema de la base de datos en la inferencia requerida, a diferencia del modelo sin ajustar que sí la necesita. Esta base en su totalidad tiene la funcionalidad de exponer todos aquellos patrones y similitudes que en el entrenamiento se puedan encontrar con el fin de especializar las respuestas a las solicitudes sobre la base de datos principal. El número de registros plasmados fue de 300, de los cuales se buscó tener la máxima variedad sintáctica, abarcando gran parte de las cláusulas, agregaciones y operadores posibles dentro de todo el conjunto de datos, así como registrar diferentes dificultades de sentencias. Se registran las frecuencias de presencia de cada clausula, agregación y operador presente en la base *gold* en el apéndice C tabla 15,16 y 17. Adicionalmente se recalca que esta base se dividió en 80% para entrenamiento,10% para validación de este y 10% para pruebas.

8.3 desarrollo e implementación de modelos en el contexto problema

Una vez terminado el proceso de limpieza, creación de los recursos necesarios y registro de los resultados preliminares, se pone en marcha la implementación de los modelos mistral-7b-v0.1 y SQLcoder-7b-2 que de forma inicial se descargan en un archivo GGUF el cual es un formato binario diseñado para almacenar y ejecutar modelos LLM optimizado para ejecución e inferencia de manera local. Este proceso se realizó mediante Visual Studio y un archivo de Python donde se cargó el modelo se realizó el *prompt* inicial el cual componía una instrucción sencilla y el esquema de base de datos del problema principal, se tomaron en cuanta los 30 registros de prueba para evaluar con las métricas establecidas y así ver el rendimiento inicial previo al ajuste por *fine-tunnig*. Se evidenciaron los siguientes resultados.

Tabla 4

Resultados Modelos sin fine-tuning sobre base del problema objetivo.

Modelo	Exact match	Execution accuracy	Error rate
Mistral	6.66%	30%	36.6%
SQLcoder	0%	16.66%	16.66%

Nota. Elaboración propia

Como principales análisis podemos contrastar lo evidenciado en los resultados preliminares debido que el modelo de mistral al tener mejores generalizaciones pudo conseguir un rendimiento más alto en la métrica de *execution accuracy*, con una diferencia de aproximadamente 5 sentencias mejor ejecutadas. Del mismo modo se pudo observar como el SQLcoder tuvo mejores rendimientos en la métrica de *error rate*, demostrando que su especialización en SQL le ayuda a generar sentencias con mejor sintaxis que mistral.

8.4 Fine-tunning

Como paso siguiente, dado que se tiene una guía de cómo fueron los resultados sin ningún ajuste sobre el modelo, se buscará encontrar la mejor combinación de parámetros que

se pueda a partir del uso de 2 tipos de parámetros, los parámetros que le pertenecen propiamente al *fine-tuning* y los que se ajustan dentro del entrenamiento. De los posibles parámetros a escoger en cada tipo se tienen una gran cantidad, por lo que al realizar las pruebas y combinaciones necesarias se obtendrían muchos casos, dificultando tanto su realización como interpretabilidad, se optó por restringir la selección a los parámetros más relevantes e influyentes en el proceso.

Para el *fine-tuning* mediante la técnica LoRA se seleccionaron los parámetros *rank* y *lora alpha*. El *rank* determina la dimensionalidad de las matrices de adaptación de bajo rango; valores más bajos reducen el número de parámetros entrenables y el costo computacional, mientras que valores más altos permiten mayor capacidad de adaptación del modelo. Por otra parte, *lora alpha* actúa como un factor de escala que controla la magnitud de las actualizaciones aplicadas a los pesos originales del modelo, siendo su relación con el *rank* lo que define el impacto real de la adaptación.

En cuanto a los parámetros de entrenamiento, se consideraron el *per device train batch size, num train epochs* y *learning rate*. El tamaño de *batch* por dispositivo afecta directamente la estabilidad del gradiente y el uso de memoria; el número de épocas determina cuántas veces el modelo recorre el conjunto de datos de entrenamiento, influyendo en el grado de ajuste alcanzado; y la tasa de aprendizaje controla la velocidad con la que el modelo actualiza sus pesos, siendo un factor crítico para lograr convergencia sin caer en sobreajuste.

Con el fin de aislar el efecto de los parámetros seleccionados y garantizar una comparación justa entre experimentos, el resto de los parámetros tanto de LoRA como del entrenamiento se mantuvieron constantes a lo largo de todas las pruebas. En la configuración de LoRA, se fijaron los módulos objetivo en las proyecciones de atención del modelo para

turnitin Página 57 de 85 - Engrega de integridad Identificador de la entrega trn:oid:::26268:603805258

los elementos de atención *key, query, values* con un *dropout* de 0.05, sin ajuste de sesgos, tipo de tarea causal, e inicialización de pesos mediante distribución gaussiana. En cuanto al entrenamiento, se mantuvieron fijos parámetros como el tamaño de *batch* de evaluación, los pasos de acumulación de gradiente el decaimiento de pesos, la norma máxima del y los pasos de calentamiento. Por último, la semilla aleatoria 42, así como las estrategias y frecuencias de registro y evaluación. Esta decisión permite atribuir las variaciones en el desempeño del modelo directamente a los parámetros que sí fueron modificados comparando los resultados obtenidos contra la línea base establecida en la sección anterior. Los resultados de cada una de las combinaciones se encuentran registradas en las tablas 11,12,13 y 14.

Dentro de las evidencias que se extrajeron del entrenamiento se registra los pasos de la combinación con mejores resultados para el modelo Mistral de la siguiente manera:

Tabla 5

Métricas de entrenamiento y validación durante el ajuste fino del modelo Mistral.

Step	Training Loss	Validation Loss	Entropy
5	1.235018	0.705916	0.690104
15	0.579157	0.563922	0.494010
25	0.475940	0.477915	0.454297
35	0.451337	0.392881	0.461849
50	0.427961	0.342582	0.335482
65	0.285260	0.348642	0.308594
80	0.293095	0.353346	0.333854
100	0.343683	0.327654	0.326823
115	0.237717	0.325071	0.257682
125	0.200194	0.309687	0.256055
140	0.186656	0.306661	0.228646
145	0.206009	0.302586	0.224414
165	0.171719	0.303741	0.226497
170	0.155830	0.305550	0.215299
180	0.198139	0.304179	0.214974

Se presenta un resumen de la evolución de las métricas obtenidas durante el proceso de ajuste fino del modelo, mostrando una selección representativa de los pasos de entrenamiento más relevantes. En términos generales, se observa una disminución progresiva de la pérdida de entrenamiento y de validación a medida que avanzan las iteraciones, pasando de valores iniciales de 1.235 y 0.706 a valores cercanos a 0.198 y 0.304, respectivamente. De igual forma, respecto a la entropía cruzada, se observó una reducción progresiva a lo largo del proceso de ajuste fino, pasando de un valor inicial de 0.690 a un valor final cercano a 0.215. Aunque se presentaron fluctuaciones menores en algunos pasos de entrenamiento, la tendencia general descendente evidencia que el modelo fue incrementando su capacidad para asignar una mayor probabilidad a los tokens correctos de las consultas objetivo

8.5 Exportación de los modelos

Una vez se obtuvo la mejor combinación para ambos modelos se realizó un proceso para poder pasar de un modelo alojado en Colab al formato gguf especializado en mejorar el rendimiento de las inferencias en un entorno local.

El punto de partida es un modelo base preentrenado, en este caso Mistral 7B, sobre el cual se aplica la técnica de adaptación LoRA. Con el fin de optimizar el consumo de memoria durante esta fase, se recurre a la cuantización en 4 bits. Esta estrategia permite mantener congelados los pesos originales del modelo y entrenar únicamente un conjunto reducido de parámetros adicionales, lo que reduce considerablemente los requisitos de memoria en GPU. Como resultado de esta etapa se obtienen los pesos del adaptador LoRA.

Una vez finalizado el entrenamiento, se almacena exclusivamente el adaptador junto con el tokenizador. Al tratarse de archivos de tamaño reducido, del orden de los megabytes, su gestión, almacenamiento y transferencia resultan sencillos.

No obstante, dado que el formato GGUF no admite adaptadores LoRA de forma directa, es necesario fusionar el adaptador con el modelo base antes de proceder a la conversión. Para ello, el modelo base se carga en precisión FP16 y se le aplican los pesos LoRA, generando un modelo completo que incorpora el conocimiento adquirido durante el *fine-tuning*. Este modelo fusionado, almacenado en formato estándar de Hugging Face, ocupa aproximadamente 14 GB.

Debido a las limitaciones de almacenamiento propias de entornos como Kaggle, el modelo fusionado se exporta hacia una plataforma con mayor capacidad, como Google Colab o un entorno local, comprimiéndolo y dividiéndolo en partes para facilitar su descarga y reconstrucción posterior.

Por último, se procede a la conversión al formato GGUF. Por razones de compatibilidad, la conversión se realiza en dos pasos: primero se genera una versión intermedia cuantizada en 8 bits y posteriormente se aplica una cuantización adicional para obtener la versión final en formato Q4_K_M. Esta configuración representa un punto de equilibrio adecuado entre la reducción del tamaño del archivo, que queda en un rango de 4 a 5 GB, la velocidad de inferencia y la preservación de la calidad del modelo.

8.6 Presentación de la interfaz

La interfaz diseñada, así como se comentó en la exploración preliminar se realizó por medio de la librería *streamlit,* la cual permite crear aplicaciones web a partir del lenguaje de programación Python de manera sencilla y completa. Para utilizar el modelo escogido se tuvo

que pasar por el proceso de exportación previamente definido y subir el archivo GGUF dentro del proyecto realizado en github, asimismo se intentó desarrollar de una forma que fuera lo mas amigable y sencilla posible, la cual se divide en 4 parte, la parte superior contiene el nombre de la herramienta, luego aparece un menú desplegable en el cual se encuentra el esquema de la base de datos, en caso de que algún usuario quiera consultar el nombre de alguna columna, en la parte intermedia se encuentra la caja donde el usuario podrá escribir su pregunta en lenguaje natural para que posteriormente de click en el botón de “generar y ejecutar SQL” el cual le pasara al modelo en local la pregunta y realizara la inferencia, en caso de que la sentencia sea correcta, aparecerá en la parte inferior acompañada del resultado después de haber sido ejecutado con la base de datos almacenada. Por último, en el costado derecho se registra un historial en el cual los usuarios podrán consultar las sentencias SQL generadas.

Figura 6

Interfaz de usuario final



Para mostrar cómo se vería la ejecución directamente desde la base de datos se muestra la siguiente figura

Figura 7

Ejecución en local de pregunta SQL en base de datos.

```python
import sqlite3
import pandas as pd

conn = sqlite3.connect("mi_base_act.db")

query = """
SELECT AVG(tasa_efectiva_promedio_ponderada) FROM Credito WHERE tipo_credito = 'consumo'
 AND tipo_tasa = 'fs';
"""

df = pd.read_sql(query, conn)
print(df)

conn.close()
```

```
   AVG(tasa_efectiva_promedio_ponderada)
0                               44.61225
```

9. Resultados

Con el fin de exponer los resultados de manera integral. Esta sección se divide en dos niveles de análisis. En primer lugar, se centrará en exponer y comparar los resultados encontrados antes y después del uso del *fine-tuning* sobre la base del problema. Como segundo aspecto se hará una revisión de los parámetros y las configuraciones realizadas, con el fin de poder identificar las razones por las cuales algunas combinaciones tuvieron un mejor comportamiento que las demás, tomando en cuenta como el cambio en los valores de cada parámetro efectúan una alteración en los resultados de las métricas.

Para cumplir con el primer nivel de análisis se presenta la siguiente tabla la cual tiene la información de media y desviación estándar muestral de las pruebas realizadas por cada modelo y cada estrategia tomada en cuenta.

Tabla 6

Resultados por modelo y estrategia (media ± desviación estándar).

Mistral			
Estrategia	Exact match	Execution accuracy	Error Rate
Prompt engineering	4.4% ± 1.5%	26.63% ± 2.73%	24.44% ± 8.74%
Sin *fine-tuning*	6.66%	22.2% ± 10.70%	26,66% ± 8.81%
Fine-tuning LoRA	**58.78% ± 5.07%**	**72.53% ± 5.66%**	**11.68% ± 5.36%**
SQLcoder			
Prompt engineering	0%	21.10% ± 6.28%	14.44% ± 3.13%
Sin *fine-tuning*	0%	17.77% ± 1.92%	18.88% ± 1.92%
Fine-tunig LoRA	27.95% ± 18.48%	43.92% ± 20.60%	40.58% ± 20.73%

Nota. Elaboración propia. El *prompt engineering* y las pruebas *sin fine-tuning* corresponden a muestras de 3 observaciones cada una. El *fine-tuning* cuenta con 48 observaciones por modelo. Todas las pruebas se realizaron bajo el mismo conjunto de datos *test* para el problema objetivo final. Las fuentes de datos y *scripts* se encuentran disponibles en el repositorio de GitHub del proyecto (Bermúdez Puentes, 2026).

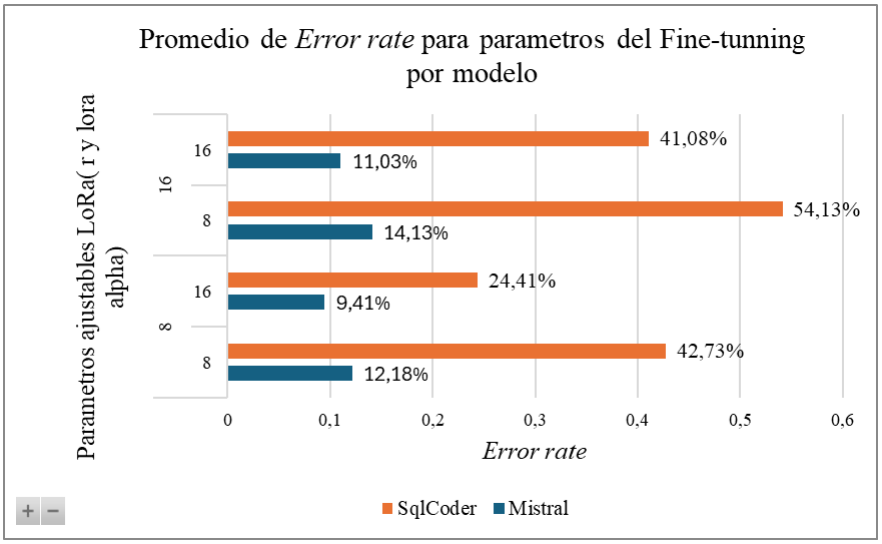
Como análisis descriptivo se puede evidenciar como los puntajes para la estrategia del *fine-tuning* en el modelo de Mistral fueron ampliamente superiores sobre el modelo SQLcoder, por lo que se puede decir que Mistral como modelo tiene una mayor capacidad de generalización para tareas de texto a SQL, incluso previo a la implementación del ajuste especializado, el cambio más drástico de resultados se da en la combinación Mistal+LoRA subiendo en la métrica de *exact match* ~5% a 58%, en cuanto a *execution accuracy* ~28% a un 72% y el *error rate* decrece de ~30% a un 11%.

Por otro lado, tanto en Mistral como en SQLcoder las diferencias entre las estrategias sin *fine-tunning* y *prompt engineering* no parecen ser muy distantes por lo que da indicios de que sin un ajuste en la estructura del modelo las mejoras se verán limitadas al no poder resolver las tareas pedidas por medio de solo instrucciones.

En cuanto a la variabilidad de las respuestas se puede ver como en Sql coder presentan unos intervalos más amplios denotando inestabilidad en el comportamiento final, en contraste Mistral en cada métrica presenta variaciones alrededor del 5% por lo que es una opción más confiable y reproducible en cuanto a resultados. Mistral con *fine-tuning* LoRA es una combinación interesante para seguirla estudiando en cuanto a la combinación más alineada. SQLcoder, a pesar de ser un modelo diseñado específicamente para SQL, no logra superar a Mistral en ningún escenario, y su comportamiento tras el *fine-tuning* es inconsistente.

Como segundo análisis entraremos a exponer a través de gráficas los diferentes rendimientos para las combinaciones de parámetros en ambos modelos. Como primera característica a tomar en cuenta se encuentra el *error rate* el cual dentro de las métricas presentes nos brinda información sobre la efectividad en producir sentencias correctamente estructuradas y en este caso la combinación de parámetros del *fine-tuning* tomados en cuenta.

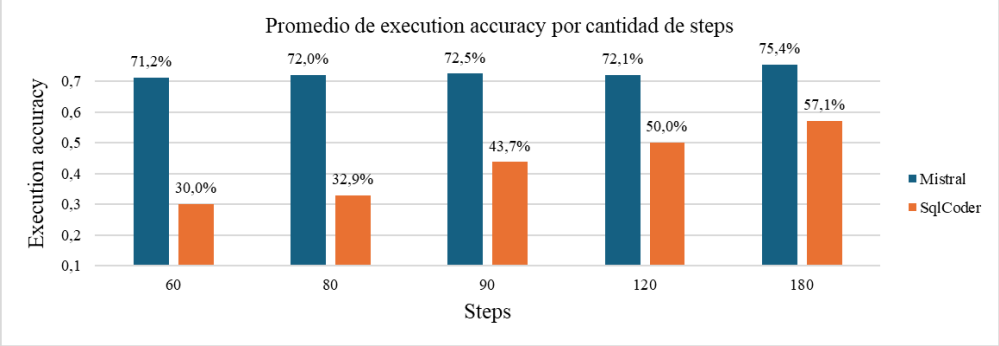
Figura 8



De la anterior figura podemos visualizar el comportamiento de ambos modelos, de acuerdo con el apartado anterior los resultados de mistral son comparablemente mejores que el SQLcoder sin embargo, resalta la combinación de parámetros c=8 y lora Alpha=16 debido a ser el valor más bajo de todos, denotando un perfil más robusto, mientras que SQLcoder pierde su particular capacidad, volviéndose más inestable y sensible a cambios en los parámetros.

Por otra parte, considerando los parámetros ajustables para el entrenamiento surge una propiedad inherente que son los *steps*, este componente no es ajustable, sino que es el resultado de la combinación de parámetros, es decir; surge como la cantidad de veces que el modelo ajusta sus pesos debido a las épocas y el tamaño de los lotes que se entrenan a la vez. De acuerdo a esto y el caso específico del problema las combinaciones que generaron 180 *steps* lograron mejorar su *execution accuracy con* cierta tendencia progresiva, sin embargo, en el momento que se evidencie una disminución se puede denotar como una alerta debido a que sobre ajusta los pesos al entrenamiento y las generalizaciones en la fase de prueba empeoran.

Figura 9



Por último, vemos tanto el comportamiento de los valores del *train batch* y *train epoch* que al analizar conjuntos de datos pequeños y más épocas el rendimiento de la métrica de *execution accuracy* incrementa, lo cual tiene mucho sentido porque el numero de *steps* aumenta. Entre más sean los conjuntos de datos analizados y mas veces se vea el *dataset* el numero de *steps* incrementara.

Figura 10

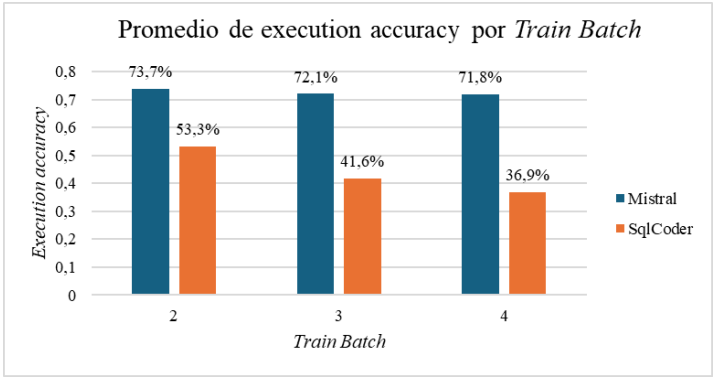
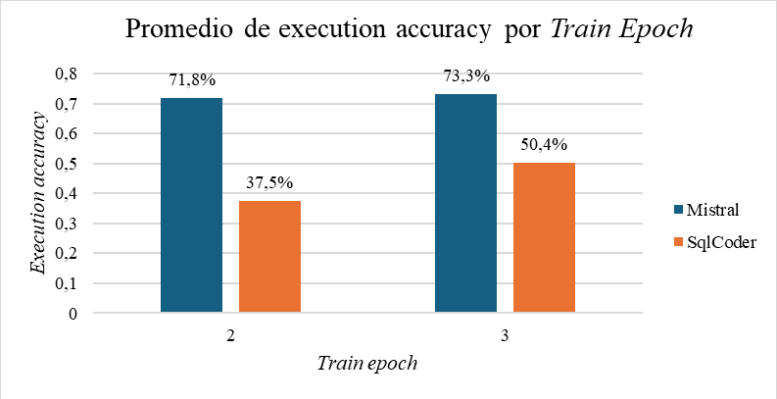


Figura 11



Por lo cual, dado los resultados mencionados se llevo al proceso de exportación del modelo e implementación en la interfaz la combinación de *finetunning* + Mistral con parámetros *r*=8, *lora Alpha*=16, t*rain batch size*=2, *train epoch*=3 y *learning rate*=2e-4. Teniendo como mejores resultados *execution accuracy* de 83,3% , *exact match* de 66,6% y *error rate* de 3,3%.

En contraste la relación con el objetivo específico asociado al desarrollo de una interfaz para la interacción con el modelo se cumplió con desarrollar una herramienta que permite a los usuarios realizar consultas en lenguaje natural y visualizar los resultados generados de manera sencilla. La interfaz fue diseñada buscando una interacción simple, requiriendo menos de tres clics para ejecutar una consulta y acceder a la respuesta generada. Adicionalmente, incorpora funcionalidades complementarias, como la consulta del historial de interacciones realizadas.

Estas características muestran un diseño orientado a facilitar la interacción con el sistema y la reducción de la complejidad para el usuario final. Sin embargo, debido a que el alcance de la investigación no contempló la realización de pruebas de usabilidad con personas, ni la implementación de una evaluación, no es posible afirmar de manera

concluyente el nivel de usabilidad de la herramienta. En consecuencia, los resultados obtenidos permiten evidenciar la usabilidad técnica de la interfaz y su potencial de utilización, mientras que la validación de aspectos relacionados con la experiencia de usuario, facilidad de aprendizaje y satisfacción queda planteada como una posibilidad de mejora en el trabajo futuro.

10. Conclusiones

En términos generales alrededor del desarrollo de la investigación realizada, se pueden elaborar diferentes reflexiones. La primera, en relación con los retos a los que se tuvieron que hacer frente para culminar los objetivos planteados, la segunda a las enseñanzas que dejó todo el ciclo del proyecto, la tercera a las posibles acciones a realizar para complementar los hallazgos encontrados, las cuales tendrán el objetivo establecer afirmaciones más contundentes para una herramienta más integral. Por último, los posibles caminos futuros en los que se puede desenvolver esta herramienta creada.

De forma inicial los resultados obtenidos permiten evidenciar el potencial de los modelos de lenguaje para facilitar el acceso a información financiera mediante consultas expresadas en lenguaje natural. Esto permitiría que un ciudadano del común formule preguntas como el monto total de créditos desembolsados en una región, los municipios con mayor acceso a financiación o la evolución de los desembolsos en determinados periodos de tiempo para entidades particulares, sin necesidad de conocimientos técnicos en bases de datos o SQL. No obstante, una solución basada únicamente en la traducción de lenguaje natural a SQL constituye un primer paso hacia la democratización del acceso a la información. Como trabajo futuro, resulta necesario complementar la herramienta con mecanismos de

explicación e interpretación de resultados que faciliten la comprensión de la información obtenida y fortalezcan su contribución a la inclusión financiera.

Adicionalmente, se considera que la herramienta diseñada fue un verdadero reto en cuanto a todos los conceptos necesarios para la comprensión y aplicación de los LLM estudiados. Poder alinear y gestionar cada paso dentro de la metodología fue un esfuerzo constante que convirtió todas las limitaciones tanto computacionales como de recursos en una guía de trabajo diaria. En segundo lugar, el proceso investigativo dejó importantes aprendizajes a la hora de la experimentación, en temas como enfoques de trabajo particularmente en *prompt engineering* y *fine-tuning*, comprendiendo su relevancia en los desempeños de cada modelo.

De igual manera se sugiere comparar el comportamiento del modelo implementado con alternativas de mayor capacidad o bajo suscripción, con el objetivo de analizar diferencias en rendimiento y escalabilidad. Se plantea la posibilidad de extender la herramienta a nuevas temáticas, ampliando su aplicabilidad en distintos dominios.

Asimismo, en cuanto a las líneas futuras de trabajo se reconocen oportunidades de mejora y profundización como evaluar el comportamiento de los *steps* durante el proceso de fine-tuning con el objetivo de encontrar puntos de deterioro en las métricas de evaluación para así identificar un techo en el desempeño de los modelos, de la misma manera se propone fortalecer el sustento estadístico mediante el incremento de muestras en cuanto al *prompt engineering* y fine-tuning para poder generar comparaciones de muestras robustas. En cuanto a la interfaz se evidencia que el diseño es muy acertado de acuerdo con lo planteado, lo cual hace que se pueda escalar esta, buscando una implementación en la web para así evaluar el

tráfico del uso de la herramienta y así orientar nuevas misiones a favor de brindar fácilmente la información financiera a los usuarios.

11. Referencias

1. Gobierno de Colombia. (2025). *Acerca de Portal de Datos Abiertos*. Datos Abiertos. Recuperado el 22 de septiembre del 2025 de https://www.datos.gov.co/stories/s/Acerca-de-Portal-de-Datos-Abiertos/rbs9-xt6j/
2. Departamento Administrativo de la Función Pública. (2015). *Decreto 103 de 2015: Por el cual se reglamenta la Ley 1712 de 2014 y se dictan otras disposiciones*. Recuperado de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=56882
3. Ministerio de Justicia y del Derecho. (s. f.). *ABC de la Ley de Transparencia y del Derecho de Acceso a la Información Pública Nacional*. Recuperado de https://www.minjusticia.gov.co/programas-co/tejiendo-justicia/Documents/publicaciones/transparencia/ABC.pdf
4. Banco Mundial. (2024). *Inclusión financiera*. Recuperado de https://www.bancomundial.org/es/topic/financialinclusion/overview
5. ProQuest. (2019). *Natural Language Processing*. Recuperado de https://www.proquest.com/openview/a44d67c88cfaada2563dc16f94ccd3c8/1?pq-origsite=gscholar&cbl=1006393
6. Ruiz, J. C., & Tabares, M. C. (2021). Procesamiento de Lenguaje Natural: aplicaciones, retos y perspectivas. *Revista Ingeniería, Matemáticas y Ciencias de la Información*, 8(16), 207–228. https://www.redalyc.org/pdf/6887/688773657016.pdf
7. IBM. (s. f.). *Natural Language Understanding*. IBM Think. Recuperado de https://www.ibm.com/think/topics/natural-language-understanding
8. Google Cloud. (2022). *Techniques for improving text-to-SQL accuracy*. Recuperado de https://cloud.google.com/blog/products/databases/techniques-for-improving-text-to-sql
9. Huandong,C. , Zicheng M. , Mingyuan M., Zhenting Q , Sabot A., Jiang H. , H. T. Kung (2025) ElaLoRA: Elastic & Learnable Low-Rank Adaptation for Efficient Model Fine-Tuning, Harvard University, SEAS.
10. Zuiderwijk, A., Janssen, M., Choenni, S., Meijer, R., & Alibaks, R. S. (2016). Open data and its usability: an empirical view from the citizen's perspective. Information Systems Frontiers, 18(2), 285–302. https://doi.org/10.1007/s10796-016-9679-1
11. Nyarko, E. S., Amoateng, K., & Aboagye, A. Q. Q. (2023). *Financial inclusion and poverty: Evidence from developing economies. International Journal of Social Economics, 50*(12), 1719–1734. https://doi.org/10.1108/IJSE-11-2021-0690
12. Segura, D. F. (2023). *Herramienta para el análisis de los metadatos del portal Open Data Colombia*. Recuperado de: http://hdl.handle.net/11522/1991
13. Z. Shaikhiyeva, M. Mansurova and G. Amirkhanova, "Text to SQL Transformation Using LLM: A Comparative Research of T5, Seq2Seq, and SQLNet Models," *2024 9th International Conference on Computer Science and Engineering (UBMK)*, Antalya, Turkiye, 2024, pp. 938-943, doi: 10.1109/UBMK63289.2024.10773579.
14. S. Karki, P. Karki, B. L. Shrestha and T. N. Jha, "Smaller Large Language Models for Text-To-Sql: Performance Analysis and Optimal Performance," *2025 International Conference on Inventive Computation Technologies (ICICT)*, Kirtipur, Nepal, 2025, pp. 1-7, doi: 10.1109/ICICT64420.2025.11005216.

15. A. A. Sai, S. Vignesh, S. H. S V and P. K. Baruah, "Comparing Accuracy and Consistency: LLMs vs. SOTA Deep Learning Models in Text-to-SQL," *2024 IEEE 31st International Conference on High Performance Computing, Data and Analytics Workshop (HiPCW)*, Bangalore, India, 2024, pp. 63-67, doi: 10.1109/HiPCW63042.2024.00020.
16. Portal de Datos Abiertos de Colombia. (s. f.). *Tasas de interés activas por tipo de crédito – Histórico*. https://www.datos.gov.co/Econom-a-y-Finanzas/Tasas-de-inter-s-activas-por-tipo-de-cr-dito-Hist-/w9zh-vetq
17. Roucher, A., Villanova del Moral, A., Wolf, T., von Werra, L., & Kaunismäki, E. (2025). *smolagents: a barebones library to build great agentic systems* [Software library]. https://github.com/huggingface/smolagents
18. Huet, P. (2023, abril 13). Qué son las redes neuronales y sus aplicaciones. *Openwebinars.net*. https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/
19. Fu, Z., Song, W., Wang, Y., Wu, X., Zheng, Y., Zhang, Y., ... & Zhao, X. (2025). Sliding window attention training for efficient large language models. *arXiv preprint arXiv:2502.18845*.
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.
21. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. Available at: https://arxiv.org/pdf/2307.09270.pdf.
22. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2005). Database Systems Concepts (5a ed.). McGraw Hill Higher Education.
23. Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., ... & Radev, D. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In Proceedings of the 2018 conference on empirical methods in natural language processing (pp. 3911-3921).
24. Shin, J., Tang, C., Mohati, T., Nayebi, M., Wang, S., & Hemmati, H. (2023). *Prompt engineering or fine-tuning: An empirical assessment of large language models for code*. arXiv. https://doi.org/10.48550/arXiv.2310.10508.
25. Bermúdez Puentes, I. (2026). *Tesis-IBP-Texto-to-SQL* [Repositorio de código]. GitHub. https://github.com/1saacbp/Tesis-IBP-Texto-to-SQL

12. Apéndice

Apéndice A: Información modelos

Tabla 7

Parámetros estructura transformer Mistral.

Parámetro	Valor	Descripción técnica
dim	4096	Dimensión del embedding. Cada token se representa como un vector en el espacio vectorial $\mathbb{R}^{4096}$.
n_layers	32	Número de bloques Transformer (capas del decodificador).
n_heads	32	Número de cabezas de atención gracias al uso del mecanismo Multi-Head Attention.
head_dim	128	Dimensión de cada cabeza de atención. Se cumple: 4096 = 32 × 128.
n_kv_heads	8	Número de cabezas para *key* y *values* (Grouped-Query Attention). Optimiza la memoria y costo computacional.
hidden_dim	14336	Dimensión interna del Feed-Forward Network.
window_size	4096	Tamaño de ventana en Sliding Window Attention. Cada token atiende solo a los 4096 tokens anteriores.
context_len	8192	Longitud máxima de contexto (pregunta + respuesta generada).
vocab_size	32000	Tamaño del vocabulario. La matriz de embeddings es 32000 × 4096.

Nota. dentro de la presente tabla se muestran los diferentes valores de los parámetros utilizados como estructura principal del *transformer*.

Tabla 8

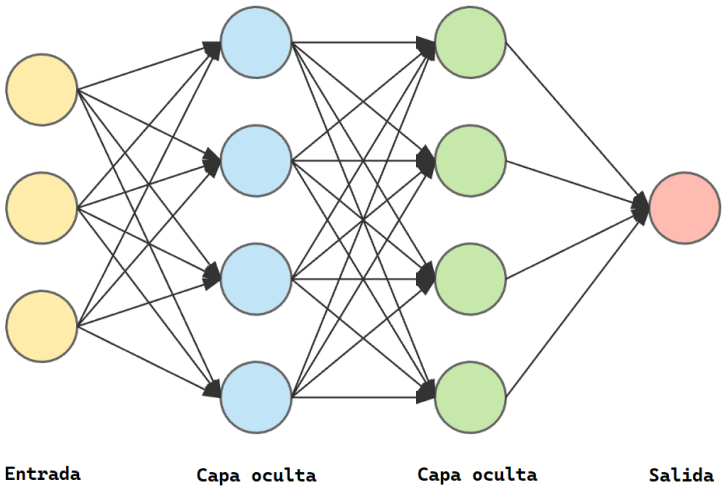
Información perteneciente a la tabla Receipts de la prueba uno.

Id_recibo	Nombre de cliente	Precio (USD)	Propina (USD)
1	Alan Payne	12.06	1.20
2	Alex Mason	23.86	0.24
3	Woodrow Wilson	53.43	5.43
4	Margaret James	21.11	1.00

Nota. Elaboración propia. Esta información fue cargada con fines académicos por lo que contiene datos ficticios que contribuyen al uso de las herramientas.

Figura 12

Estructura básica de una red neuronal.



Nota. De "*Qué son las redes neuronales y sus aplicaciones"* por Pablo Hueta, 2023(https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/)

37

Apéndice B: resultados pruebas iniciales

Tabla 9

Presentación de Sentencias generadas por modelos LLM con observaciones.

Preguntas	Observaciones	Resultado SQLcoder	Resultado Mistral base
¿Cuáles son los nombres y edades de todos los clientes registrados?	El modelo SQLcoder hace uso de alias lo que hace resultados más limpios	SELECT c.name, c.age FROM customers c;	SELECT c.name, c.age FROM customers c JOIN products_orders po ON c.customer_id = po.customer_id;
¿Qué productos ha comprado cada cliente y en qué cantidad?	Ambos modelos lo hicieron correctamente	SELECT c.name, p.name, po.quantity FROM customers c JOIN products_orders po ON c.customer_id = po.customer_id JOIN products p ON po.product_id = p.product_id;	SELECT c.name, p.name, p.price, q.quantity FROM customers c JOIN products_orders q ON c.customer_id = q.customer_id JOIN products p ON p.product_id = q.product_id
¿Cuánto dinero total ha gastado cada cliente en compras?	El modelo Sqlcoder no pudo realizar correctamente la agregación del precio por las cantidades además se recalca que el mistral hace un ordenamiento de mayor a menor gasto.	SELECT c.name, SUM(p.price) AS total_spent FROM customers c JOIN products_orders po ON c.customer_id = po.customer_id JOIN products p ON po.product_id = p.product_id GROUP BY c.name;	SELECT customers.name, SUM(products_orders.quantity * products.price) AS total_spent FROM customers INNER JOIN products_orders ON customers.customer_id = products_orders.customer_id INNER JOIN products ON products_orders.product_id = products.product_id GROUP BY customers.name ORDER BY total_spent DESC;
¿Cuál es el cliente que ha gastado más dinero en total y cuánto ha gastado?	El modelo Sqlcoder no pudo realizar correctamente la agregación del precio por las cantidades	SELECT c.name, SUM(p.price) AS total_spent FROM customers c JOIN products_orders po ON c.customer_id = po.customer_id JOIN products p ON po.product_id = p.product_id GROUP BY c.name ORDER BY total_spent DESC LIMIT 1;	SELECT c.name, SUM(p.price * p.quantity) AS total_spent FROM customers c JOIN products_orders po ON c.customer_id = po.customer_id JOIN products p ON p.product_id = po.product_id GROUP BY c.name ORDER BY total_spent DESC LIMIT 1;

Nota. Elaboración propia.

Tabla 10

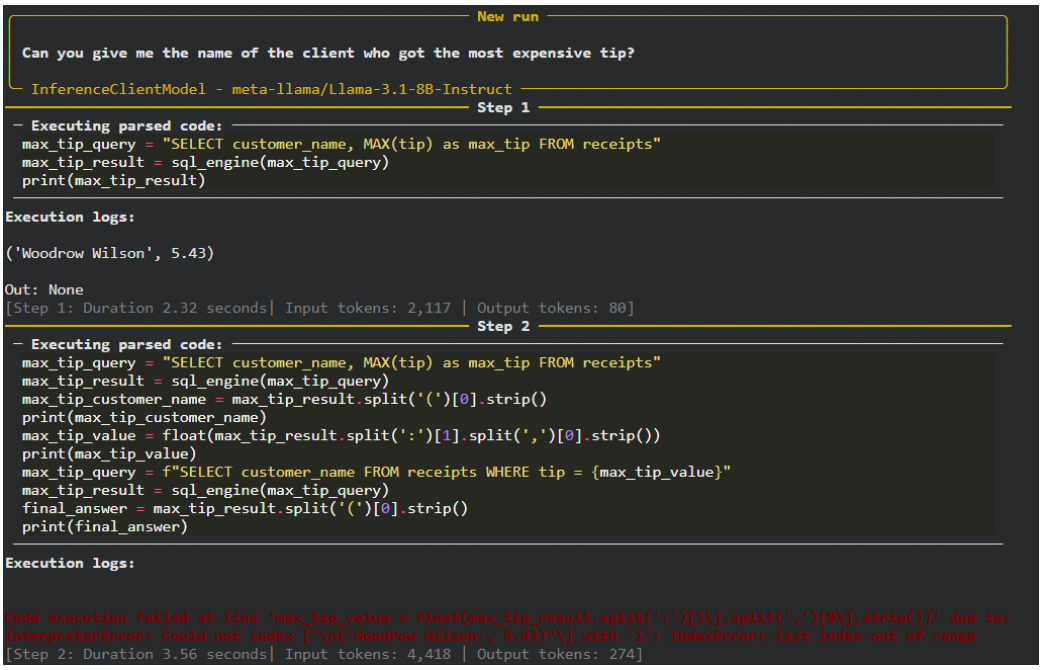
Cuadro comparativo por clausulas para cada modelo.

Pregunta	Modelo	Clausula Select y from	SQL correcto	Usa Join correcto	Usa group by correcto	Usa agregaciones correcto
P1	SQLcoder	Si	Si	N/A	N/A	N/A
P1	Mistral	Si	Si	Si	N/A	N/A
P2	SQLcoder	Si	Si	Si	N/A	N/A
P2	Mistral	Si	Si	Si	N/A	N/A
P3	SQLcoder	Si	No	Si	Si	No
P3	Mistral	Si	Si	Si	Si	Si
P4	SQLcoder	Si	No	Si	Si	No
P4	Mistral	Si	Si	Si	Si	Si

Nota. Elaboración propia.

Figura 13

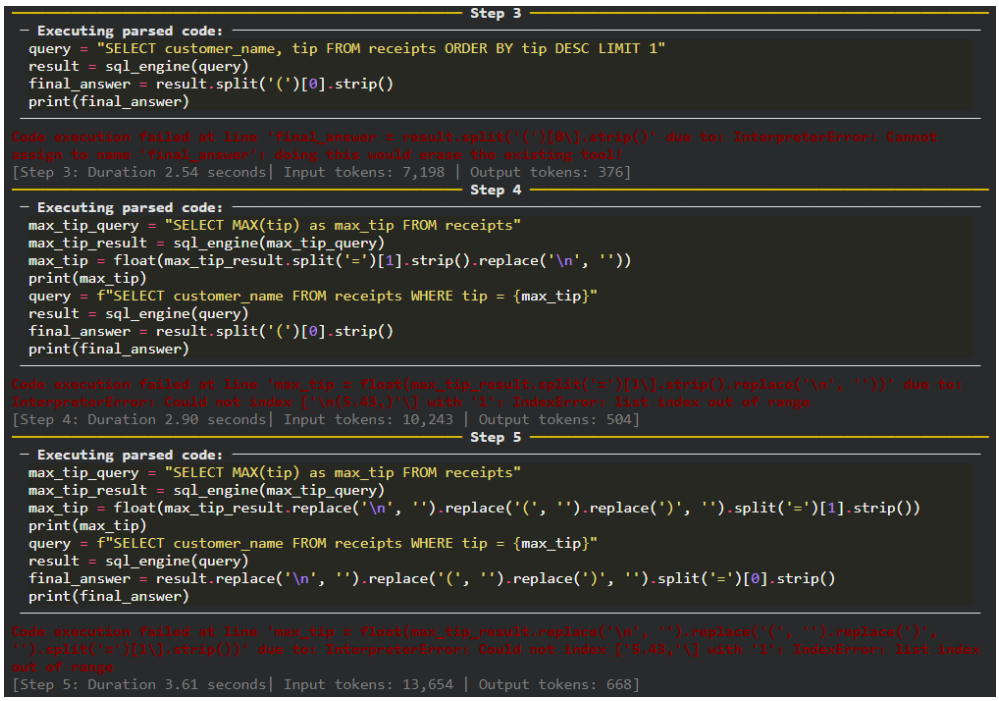
Trazabilidad del resultado obtenido por medio del agente



Nota. Pasos 1 y 2 de la respuesta de la consola.

Figura 13.1

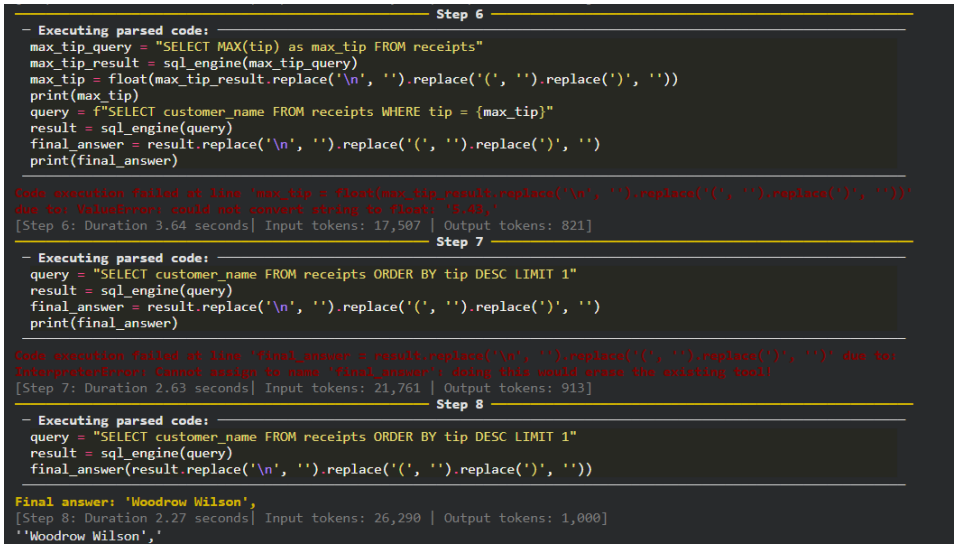
Resultados obtenidos pasos del tres al cinco



Nota. Resultados del paso tres al cinco.

Figura 13.2

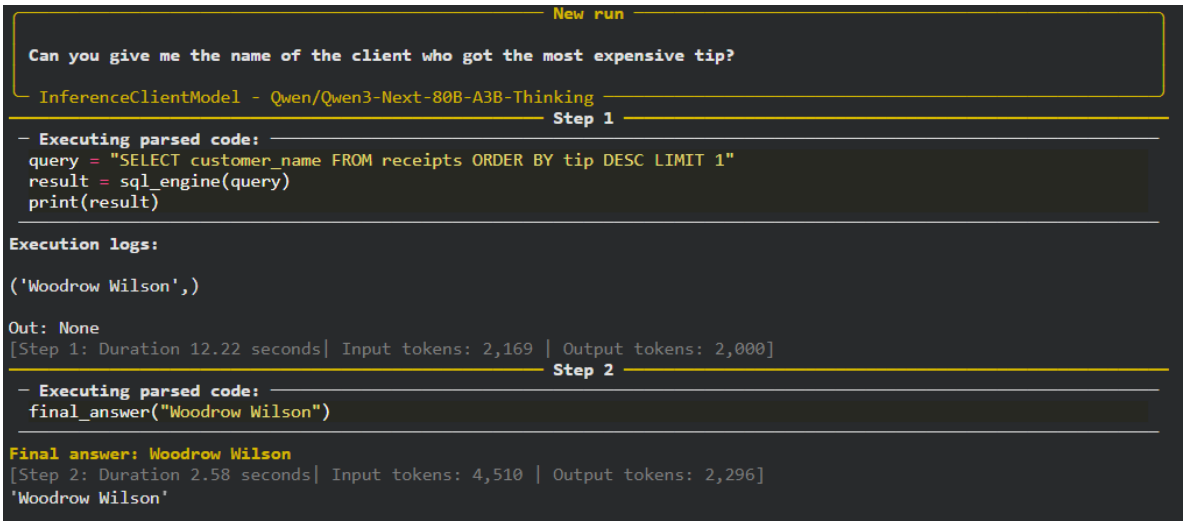
Resultados obtenidos pasos del seis al ocho



Nota. Resultados en los pasos del seis al ocho.

Figura 14

Resultado obtenido por el modelo Qwen3-Next-80B-A3B-thinking



Nota. Se presenta el resultado de consola obtenido al correr el modelo Qwen3-Next-80B-A3B-thinking.

Apéndice C: resultados problema objetivo

C.1 Prompt versión 1 (V1)

Genera la consulta SQL correcta para la siguiente pregunta.

C.2 Prompt versión 2 (V2)

Eres un experto en SQL. Genera la consulta SQL correcta para la siguiente pregunta, haz uso de la siguiente información:

Reglas:

- Usa SQL estándar.

- Usa DATE() cuando compares fechas.

- No inventes columnas.

- Usa nombres exactos de tablas.

- Usa el esquema de la base de datos presentado a continuación.

A.3 Prompt versión 3 (V3)

Eres un experto en SQL para SQLite. Dada una pregunta en lenguaje natural, genera ÚNICAMENTE la consulta SQL correcta, sin explicaciones ni texto adicional.

Reglas obligatorias:

- Usa CAST(fecha_corte AS DATE) o DATE(fecha_corte) para comparar fechas, ya que fecha_corte es TEXT.

- No uses columnas que no existan en el schema.

- Para cruzar Credito con Municipio: Credito → Deudor → Municipio.

- Para cruzar Credito con actividad económica: Credito → Deudor → Ciiu.

- Para cruzar Credito con entidad: Credito → Entidades vía id_entidad.

- SUM(monto_desembolsado) para totales, COUNT(*) o SUM(numero_creditos_desembolsados) para conteos.

- Cuando filtres por tipo_entidad, tipo_credito, tipo_persona u otros campos categóricos, usa el valor exacto como aparece en los datos.

Figura 15

Interfaz preliminar creada por medio Streamlit.



Nota. se presenta una interfaz desarrollada en la fase preliminar del proyecto, creada a partir de la librería *streamlit.*

Tabla 11

Resultados combinaciones fine-tuning modelo Mistral Pt1.

r	Lora_apha	Train batch	Train_Epoch	steps	Learning_rate	Exact match	Execution_accuracy	Error rate
8	8	2	2	120	1e-4	56,6%	73,3%	13,3%
8	8	2	2	120	2e-4	63,3%	80%	3,3%
8	8	2	3	180	1e-4	56,6%	66,6%	20%
8	8	2	3	180	2e-4	53,3%	66,6%	3,3%
8	8	3	2	80	1e-4	60%	73,3%	16,6%
8	8	3	2	80	2e-4	56,6%	66,6%	13,3%
8	8	3	3	120	1e-4	53,3%	66,6%	16,6%
8	8	3	3	120	2e-4	60%	76,6%	10%
8	8	4	2	60	1e-4	60%	70%	16,6%
8	8	4	2	60	2e-4	63,3%	76,6%	10%
8	8	4	3	90	1e-4	60%	76,6%	6,6%
8	8	4	3	90	2e-4	53,3%	70%	16,6%
8	16	2	2	120	1e-4	56,6%	70%	6,6%
8	16	2	2	120	2e-4	53,3%	63,3%	10%
8	16	2	3	180	1e-4	63,3%	80%	3,3%
8	16	2	3	180	2e-4	66,6%	83,3%	3,3%
8	16	3	2	80	1e-4	70%	80%	10%
8	16	3	2	80	2e-4	63,3%	76,6%	13,3%
8	16	3	3	120	1e-4	56,6%	70%	20%
8	16	3	3	120	2e-4	63,3%	70%	13,3%
8	16	4	2	60	1e-4	66,6%	76,6%	6,6%
8	16	4	2	60	2e-4	56,6%	76,6%	16,6%
8	16	4	3	90	1e-4	56,6%	70%	6,6%
8	16	4	3	90	2e-4	66,6%	76,6%	3,3%
16	8	2	2	120	1e-4	63,3%	76,6%	6,6%
16	8	2	2	120	2e-4	56,6%	66,6%	20%

Tabla 12

Resultados combinaciones fine-tuning modelo Mistral Pt2.

r	Lora_apha	Train batch	Train_Epoch	steps	Learning_rate	Exact match	Execution_accuracy	Error rate
16	8	2	3	180	1e-4	50%	70%	16,6%
16	8	2	3	180	2e-4	60%	76,6%	10%
16	8	3	2	80	1e-4	56,6%	63,3%	16,6%
16	8	3	2	80	2e-4	60%	73,3%	16,6%
16	8	3	3	120	1e-4	60%	73,3%	16,6%
16	8	3	3	120	2e-4	53,3%	70%	10%
16	8	4	2	60	1e-4	46,6%	56,6%	26,6%
16	8	4	2	60	2e-4	53,3%	63,3%	13,3%
16	8	4	3	90	1e-4	56,6%	70%	10%
16	8	4	3	90	2e-4	66,6%	80%	6,6%
16	16	2	2	120	1e-4	60%	76,6%	10%
16	16	2	2	120	2e-4	63,3%	70%	10%
16	16	2	3	180	1e-4	63,3%	80%	6,6%
16	16	2	3	180	2e-4	63,3%	80%	6,6%
16	16	3	2	80	1e-4	60%	76,6%	13,3%
16	16	3	2	80	2e-4	53,3%	66,6%	20%
16	16	3	3	120	1e-4	53,3%	70%	10%
16	16	3	3	120	2e-4	63,3%	80%	10%
16	16	4	2	60	1e-4	63,3%	76,6%	6%
16	16	4	2	60	2e-4	56,6%	73,3%	13,3%
16	16	4	3	90	1e-4	53,3%	66,6%	16,6%
16	16	4	3	90	2e-4	50%	70%	10%

Tabla 13

Resultados combinaciones fine-tuning modelo SQLcoder Pt1.

r	Lora_apha	Train batch	Train_Epoch	steps	Learning_rate	Exact match	Execution_accuracy	Error rate
8	8	2	2	120	1e-4	16,6%	33,3%	53,3%
8	8	2	2	120	2e-4	40%	56,6%	36,6%
8	8	2	3	180	1e-4	20%	50%	33,3%
8	8	2	3	180	2e-4	53,3%	63,3%	13,3%
8	8	3	2	80	1e-4	3,3%	16,6%	63,3%
8	8	3	2	80	2e-4	26,6%	56,6%	30%
8	8	3	3	120	1e-4	13,3%	36,6%	43,3%
8	8	3	3	120	2e-4	46,6%	56,6%	33,3%
8	8	4	2	60	1e-4	3,3%	10%	76,6%
8	8	4	2	60	2e-4	23,3%	43,3%	36,6%
8	8	4	3	90	1e-4	10%	23,3%	56,6%
8	8	4	3	90	2e-4	46,6%	56,6%	36,6%
8	16	2	2	120	1e-4	40%	56,6%	33,3%
8	16	2	2	120	2e-4	53,3%	73,3%	13,3%
8	16	2	3	180	1e-4	56,6%	66,6%	16,6%
8	16	2	3	180	2e-4	56,6%	73,3%	13,3%
8	16	3	2	80	1e-4	26,6%	50%	30%
8	16	3	2	80	2e-4	43,3%	56,6%	26,6%
8	16	3	3	120	1e-4	50%	60%	26,6%
8	16	3	3	120	2e-4	50%	73,3%	16,6%
8	16	4	2	60	1e-4	26,6%	43,3%	43,3%
8	16	4	2	60	2e-4	40%	53,3%	33,3%
8	16	4	3	90	1e-4	40%	53,3%	30%
8	16	4	3	90	2e-4	56,6%	70%	10%
16	8	2	2	120	1e-4	50%	66,6%	13,3%
16	8	2	2	120	2e-4	0%	10%	76,6%

Tabla 14

Resultados combinaciones fine-tuning modelo SQLcoder Pt2.

r	Lora_apha	Train batch	Train_Epoch	steps	Learning_rate	Exact match	Execution_accuracy	Error rate
16	8	2	3	180	1e-4	13,3%	33,3%	46,6%
16	8	2	3	180	2e-4	33,3%	53,3%	33,3%
16	8	3	2	80	1e-4	0%	3,3%	90%
16	8	3	2	80	2e-4	20%	43,3%	40%
16	8	3	3	120	1e-4	3%	13,3%	63,3%
16	8	3	3	120	2e-4	26,6%	53,3%	30%
16	8	4	2	60	1e-4	0%	0%	90%
16	8	4	2	60	2e-4	10%	23,3%	53,3%
16	8	4	3	90	1e-4	0%	10%	76,6%
16	8	4	3	90	2e-4	23,3%	46,6%	36,6%
16	16	2	2	120	1e-4	23,3%	40%	43,3%
16	16	2	2	120	2e-4	46,6%	60%	30%
16	16	2	3	180	1e-4	26,6%	53,3%	36,6%
16	16	2	3	180	2e-4	50%	63,3%	13,3%
16	16	3	2	80	1e-4	6,6%	16,6%	63,3%
16	16	3	2	80	2e-4	10%	20%	63,3%
16	16	3	3	120	1e-4	23,3%	46,6%	36,6%
16	16	3	3	120	2e-4	53,3%	63,3%	20%
16	16	4	2	60	1e-4	3,3%	10%	76,6%
16	16	4	2	60	2e-4	30%	56,6%	30%
16	16	4	3	90	1e-4	10%	30%	50%
16	16	4	3	90	2e-4	36,6%	60%	30%

Tabla 15

Presencia de cláusulas dentro de la base gold para el problema objetivo en orden descendente por frecuencia.

Cláusula	Frecuencia
JOIN	262
GROUP BY	165
WHERE	146
ORDER BY	75
LIMIT	70
HAVING	14

Nota. Elaboración propia.

Tabla 16

Presencia de agregaciones dentro de la base gold para el problema objetivo en orden descendente por frecuencia.

Agregaciones	Frecuencia
COUNT	94
AVG	85
SUM	46
MAX	14
MIN	10

Nota. Elaboración propia.

Tabla 17

Presencia de operadores dentro de la base gold para el problema objetivo en orden descendente por frecuencia.

Operadores	Frecuencia
=	412
DESC	64
AND	30
>	16
ASC	11
LIKE	9
<	7
IN	4
BETWEEN	2
OR	1

Nota. Elaboración propia.