ISBN: 978-628-96613-3-0. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2026.1.1.800

CHIMLE: An LLM-based chatbot for managing student academic databases

Paul Tocto[1], Gloria Teresita Huamaní[2], Adal Aranda[3], Diogo Abregú[4]
[1,2,3,4]Universidad Nacional de Ingeniería, Perú, ptocto@uni.edu.pe, gloria.huamani.h@uni.edu.pe, adal.aranda.n@uni.pe, diogo.abregu.g@uni.pe

Abstract– ***This paper presents CHIMLE, an intelligent chatbot prototype based on a Large Language Model (LLM) designed to interact directly with institutional student academic databases. The proposed system supports natural language-based access to academic information, automated and personalized report generation, responses to frequently asked questions, and assistance with administrative processes. Its objective is to enhance accessibility, usability, and operational efficiency in academic information management within higher education institutions. The solution is implemented in Google Colaboratory integrating a generative language model with SQL-based tools that allow secure and dynamic interaction with structured institutional data. The architecture supports text-to-SQL translation, multi-turn conversational queries, and real-time data retrieval. The prototype was evaluated through a set of representative academic queries related to student performance, course information, and aggregated statistics. The results demonstrate that CHIMLE can accurately interpret user intentions and generate reliable responses with low execution time, highlighting its potential as a flexible and scalable support tool for academic environments, as a discussion in relation to recent advances in LLM-based educational chatbots.***

Keywords-- ***LLM, Academic Chatbots; Text-to-SQL; Educational Information Management; Google Colaboratory; Higher Education***

CHIMLE: Un chatbot basado en LLM para gestionar bases de datos académicas estudiantiles

Paul Tocto[1], Gloria Teresita Huamaní[2], Adal Aranda, alumno[3], Diogo Abregú, alumno[4]
[1,2,3,4]Universidad Nacional de Ingeniería, Perú, ptocto@uni.edu.pe, gloria.huamani.h@uni.edu.pe, *adal.aranda.n@uni.pe, diogo.abregu.g@uni.pe*

Resumen*– *Presentamos CHIMLE, un prototipo de chatbot inteligente basado en un Modelo de Lenguaje Extenso (LLM), diseñado para interactuar directamente con bases de datos institucionales de estudiantes. El sistema propuesto permite el acceso a información académica mediante lenguaje natural, la generación automatizada y personalizada de reportes, la atención de preguntas frecuentes y el apoyo a procesos administrativos, con el objetivo de mejorar la accesibilidad, la usabilidad y la eficiencia operativa en la gestión de información académica en instituciones de educación superior. La solución propuesta se implementa en Google Colaboratory integrando un modelo de lenguaje generativo con herramientas basadas en SQL que permiten una interacción segura y dinámica con datos institucionales estructurados. La arquitectura admite la traducción de texto a SQL, consultas conversacionales multi-turno y la recuperación de datos en tiempo real. El prototipo se evaluó mediante un conjunto de consultas académicas representativas relacionadas con el rendimiento estudiantil, se muestra la información del curso y estadísticas agregadas. Los resultados demuestran que CHIMLE puede interpretar con precisión las intenciones del usuario y generar respuestas confiables con un bajo tiempo de ejecución, destacando su potencial como una herramienta de soporte flexible y escalable para entornos académicos, así como una discusión ampliada en relación con los avances recientes en chatbots educativos basados en LLM.

Palabras clave: Chatbots Académicos; Text-to-SQL; Gestión de Información Educativa; Google Colaboratory; Educación Superior

I. Introducción

En los últimos años, los Modelos de Lenguaje Extensos, *Large Language Model (LLM)*, como GPT-4 han transformado la interacción con datos, permitiendo interpretación y generación de texto con un alto grado de precisión. En el sector educativo, plataformas como Gradescope y los sistemas de gestión de aprendizaje, *Learning Management Systems* (LMS) han incorporado sistemas automáticos para evaluación y consulta. Sin embargo, estos sistemas no ofrecen interacción flexible en lenguaje natural, ni se integran directamente con bases de datos de alumnos. Por ello, CHIMLE busca cerrar esta brecha, proporcionando una solución ágil, accesible y adaptativa que optimiza la gestión de información académica y administrativa mediante el uso de tecnologías emergentes.

En el contexto del proyecto **CHIMLE**, VALOR-EVAL sirve como inspiración metodológica para evaluar el rendimiento del sistema, especialmente al momento de interpretar datos estructurados (como bases de datos académicas) y responder en lenguaje natural, asegurando precisión y exhaustividad [1]. Su metodología basada en dos etapas, aprovechando LLMs como GPT-4, permite evaluar modelos en tareas de lenguaje-visual, extrapolando sus principios a chatbots académicos. La capacidad de estos modelos para interpretar relaciones complejas entre conceptos es un punto de partida prometedor para gestionar bases de datos educativas, asegurando que las consultas realizadas por los estudiantes sean tanto precisas como exhaustivas.

Además, aplicar este enfoque en un chatbot académico puede ayudar a personalizar la interacción y garantizar respuestas ajustadas a las necesidades de los usuarios. Al integrar herramientas como VALOR-EVAL, se pueden mitigar los riesgos de imprecisión, optimizando el diseño de sistemas para entornos educativos basados en IA. [1] Asimismo "…mitigar los riesgos de imprecisión y garantizar la confiabilidad de las respuestas generadas por sistemas conversacionales basados en LLM, especialmente en contextos educativos [2]."

Por otra parte, el enfoque educativo y de gestión del chatbot, la adaptación culturalmente adecuada o específica al contexto y la personalización en diferentes entornos educativos. la interacción con los usuarios es factible por el Procesamiento del Lenguaje Natural (NLP) y la comprensión del Lenguaje Natural (NLU) [3] En este sentido, investigaciones recientes destacan que la interacción con chatbots basados en LLM no depende únicamente de su desempeño técnico, sino también de la forma en que los usuarios construyen socialmente su significado, rol y nivel de confianza dentro de un entorno institucional. [4]

Es altamente relevante para este proyecto, el uso de chatbots académicos basados en LLMs para gestionar información en entornos educativos y tomar como referente, especialmente sobre escalabilidad y flexibilidad, por ejemplo, el enfoque del modelo Aero Bert, este destaca los LLMs para la gestión de documentación técnica en la industria aeroespacial lo que permite resolver problemas específicos. El modelo, derivado de BERT, ha sido entrenado con millones de

documentos técnicos, lo que demuestra la capacidad de los LLMs para manejar y optimizar información técnica específica de un campo. [5]
Asimismo considerar el desarrollo y perfeccionamiento de grandes modelos lingüísticos (LLM) que tengan acceso a la comunicación digital como menciona [6] .Adicionalmente, los modelos de LLM están transformando múltiples dominios, aquí se menciona 3 casos : a) Una solución efectiva y personalizada en la corrección de pronunciación, se presenta un sistema innovador que combina Reinforcement Learning (RL) y Large Language Models (LLMs) para mejorar la corrección de pronunciación de manera personalizada, destacando su aplicabilidad en terapia del habla y aprendizaje de idiomas. [7] b). En química, biología se ha desarrollado PowerNovo, herramienta innovadora, para la secuenciación de péptidos de novo basada en el uso de datos obtenidos mediante espectrometría de masas en tándem (MS/MS) y algoritmos avanzados de inteligencia artificial, específicamente modelos de lenguaje extensos (LLMs) como los transformadores BERT [8]. c) TCMChat constituye un antecedente relevante al demostrar la viabilidad de aplicar modelos de lenguaje extensos en un dominio especializado como la medicina tradicional china, donde la precisión y confiabilidad de las respuestas son críticas. Este enfoque refuerza la posibilidad de trasladar soluciones basadas en LLM a otros contextos estructurados, como la gestión de información académica institucional [9]

No obstante, el uso de modelos de lenguaje de gran escala implica desafíos asociados al consumo intensivo de recursos computacionales, la interpretabilidad y la confiabilidad de las respuestas generadas. Por ejemplo, modelos como GPT-4 cuentan con miles de millones de parámetros [10], lo que hace necesario explorar estrategias complementarias como el aprendizaje federado y de borde para garantizar privacidad, eficiencia y escalabilidad [11]. Google Colaboratory (Colab) es una plataforma basada en la nube que permite escribir y ejecutar código Python directamente desde un navegador. Está orientada a la experimentación y desarrollo de proyectos de machine learning, deep learning y análisis de datos, ofreciendo acceso a GPUs y TPUs de manera gratuita o con un costo reducido.

En este contexto, el presente estudio responde a la pregunta; ¿Cómo es posible desarrollar un prototipo de chatbot basado en LLM, implementado en Google Colaboratory que interactúe con bases de datos de estudiantes para permitir consultas precisas, generación de reportes personalizados y automatización de tareas administrativas, mejorando la accesibilidad y eficiencia en la gestión de información académica en instituciones educativas? Y el objetivo es desarrollar un prototipo de chatbot basado en LLM en Google Colaboratory que permita la interacción en lenguaje natural con la base de datos académicas. Para ello, se aborda el diseño de la arquitectura del sistema, la identificación de las herramientas necesarias, la implementación del prototipo y la realización de pruebas funcionales que validen su desempeño en un entorno educativo real.

A. Metodología

La metodología ver Figura 1, contempla: Definir la arquitectura, Diseñar la base de datos, Construir la base de datos en SQLite en Colaboratory, Configuración de un modelo de alto rendimiento (LLM) usando Colaboratory, y Realizar pruebas

Fig. 1 Metodología

B. Proceso automatizado de las consultas [Text-to-SQL]

Text-to-SQL [12] se refiere al proceso automatizado por el cual consultas en lenguaje natural son traducidas a sentencias SQL ejecutables sobre una base de datos. El enfoque de Intelli-Dispatch-SQL integra un Modelo de Lenguaje Extenso (LLM) con módulos auxiliares específicos (reconocimiento de intención y validación de SQL), permitiendo que usuarios sin conocimientos de programación realicen consultas complejas a bases de datos solo usando lenguaje natural.

A diferencia de enfoques tradicionales basados en plantillas o reglas, el sistema aprovecha la capacidad de los LLM para comprender la semántica y contexto del usuario, generando SQL preciso incluso en consultas complejas y dinámicas, típicas de dominios críticos como el despacho energético. El flujo consiste en: analizar la intención del usuario, generar el SQL vía LLM y validar esa consulta sintáctica y semánticamente antes de ejecutarla sobre los

datos, para máxima seguridad y robustez.

Este concepto es muy importante e interesante para el desarrollo de la investigación, ya que el desarrollo de un chat inteligente para consultar bases de datos académicas se basa exactamente en este paradigma. Sus aportes clave son:

- Accesibilidad: Permite que profesores, administrativos y eventualmente estudiantes consulten información académica compleja sin saber SQL (por ejemplo, "¿Cuántos alumnos reprobaron el curso X este semestre?"), lo que democratiza el acceso a la analítica académica.
- Adaptabilidad: El marco descrito puede adaptarse fácilmente a bases de datos académicas ajustando la fase de reconocimiento de intención a los términos y necesidades del área educativa.
- Precisión y Seguridad: El módulo de validación de SQL es esencial para garantizar que las consultas generadas no sean solo sintácticamente correctas, sino también seguras y pertinentes, protegiendo la integridad de los datos académicos y evitando errores críticos.
- Flexibilidad en consultas: Habilita interacciones conversacionales multi-turno, lo que mejora la experiencia y profundidad de análisis: por ejemplo, encadenar consultas o refinar resultados a través del chat.

C. Potenciación del conocimiento/ Knowledge Augmentation

En el contexto de desarrollar un chat inteligente para consulta académica, requiere la integración LLM-Knowledge, El objetivo es combinar la capacidad generativa y el procesamiento en lenguaje natural de los LLM con la precisión y el razonamiento lógico que aportan las estructuras formales de conocimiento. Un análisis exhaustivo de las estrategias y arquitecturas para integrar Modelos de Lenguaje Extensos (LLM) con métodos basados en conocimiento estructurado, como bases de conocimiento (KB), grafos de conocimiento (KG) y sistemas híbridos como RAG. [13]. Esto permite combinar el poder conversacional de los modelos de lenguaje con el acceso y razonamiento sobre datos académicos estructurados (como registros de alumnos, relaciones entre cursos y profesores).

- Facilita que el chatbot responda no solo con texto generado o inferido, sino sustentando sus respuestas en datos reales y estructurados, consultando directamente bases o grafos relativos al dominio académico.
- Posibilita consultas complejas y multifacéticas ("¿Qué alumnos han cursado asignaturas con X profesor y aprobado todas en los últimos dos años?") de manera conversacional y precisa.
- Permite sumar técnicas como RAG, que enriquecen las respuestas recuperando evidencia o datos relevantes, y adaptarse dinámicamente a bases de conocimiento actualizadas.
- El análisis detallado del paper sobre ventajas, desafíos y estrategias es directamente aplicable para definir la arquitectura óptima y justificar tecnológicamente el desarrollo de sistemas conversacionales académicos avanzados, robustos y escalables.

A continuación, definimos la arquitectura, luego desarrollamos el prototipo en Colaboratory y realizamos pruebas del prototipo al cual le denominaremos CHIMLE (***Chat inteligente basado en un Modelo de Lenguaje Extenso (LLM)***.
Seguimos desarrollando prototipos y modelos [14 - 17] gestando una base de conocimiento para elaborar una propuesta de arquitectura de un modelo de recomendación y adaptación en un contexto educativo de educación superior en ingeniería.

II. ARQUITECTURA CHIMLE

La arquitectura de CHIMLE, ver Fig. 2, comprende los siguientes componentes, y la relación entre ellos:

Usuario: Inicia la interacción con el sistema mediante preguntas o consultas en lenguaje natural

Google Colaboratory: Entorno en la nube que permite ejecutar y alojar el sistema. Conocido como Colab, es una plataforma gratuita basada en la nube proporcionada por Google que permite a los usuarios escribir y ejecutar código Python en un entorno interactivo similar a Jupyter Notebook.

Chatbot: Interpreta las consultas y genera comandos SQL necesarios.

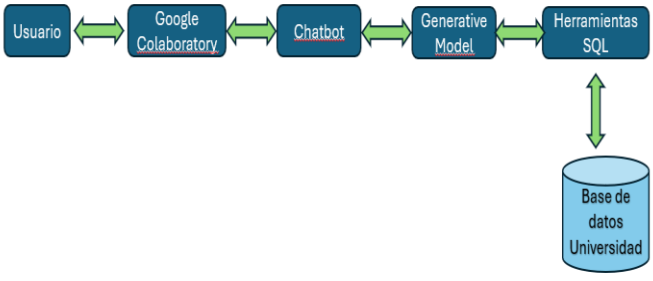

Fig. 2 Arquitectura de CHIMLE

Modelo Generativo (LLM) ***GenerativeModel*****:** procesa el lenguaje natural del usuario e interactúa con las herramientas SQL para generar respuestas precisas.

Herramientas SQL: Incluye funciones específicas como *list_tables* (listar tablas), *describe_table* (describir su estructura) y *execute_query* (ejecutar consultas) para manejar la base de datos SQL.

Base de Datos (Universidad): Base de datos SQL de la universidad. Almacena información de estudiantes, docentes, cursos, inscripciones y notas, que contiene la información requerida.

2.1 Diseño de la base de datos

Las entidades ver Fig. 3, consideradas en CHIMLE son las siguientes: Estudiante, Docentes, Cursos, Inscripciones y notas.

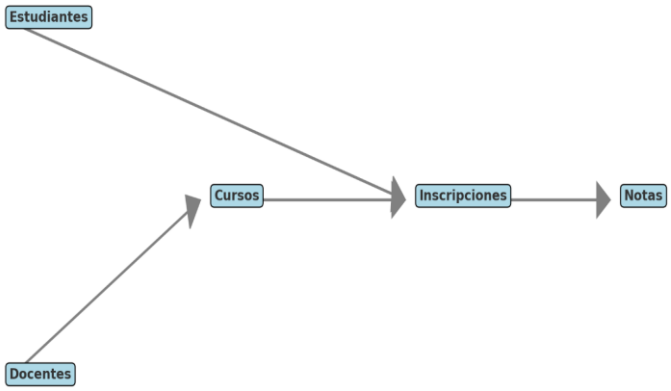

Fig. 3 Diagrama de Entidad relación de CHIMLE

2.2 Herramientas SQL

Las herramientas implementadas en CHIMLE son las siguientes: listar las tablas, describir las tablas y ejecutar una consulta.

2.3 Generative Model

El modelo que usa CHIMLE es gemini-1.5-flash-latest[18], es una variante de la familia de modelos Gemini desarrollada por Google DeepMind. Esta versión, conocida como Gemini 1.5 Flash, está optimizada para tareas de alto volumen y alta frecuencia, ofreciendo respuestas rápidas y eficientes. Se diseñó específicamente para abordar las necesidades de los desarrolladores que requieren menor latencia y costos reducidos.

2.4 Chatbot

El chat que usa CHIMLE, es una instancia de un objeto de chat a partir del modelo generativo definido en 2.3. Este objeto de chat permite interactuar con el modelo de manera continua, manejando preguntas, respuestas y posibles llamadas a funciones según sea necesario. El flujo del chat es el siguiente:

Usuario realiza una consulta:

- Por ejemplo: "¿Qué cursos están disponibles este semestre?"

El modelo analiza la consulta:

- Identifica que necesita realizar una consulta SQL para obtener la información solicitada.
- Selecciona automáticamente las funciones necesarias definidas en 2.2:
  - Usa listar tablas para verificar la existencia de la tabla Cursos.
  - Usa describir las tablas para obtener su esquema.
  - Genera y ejecuta una consulta SQL con ejecutar una consulta.

Respuesta al usuario:

- Una vez que el modelo obtiene los datos, los procesa y responde al usuario en lenguaje natural.

III. IMPLEMENTACIÓN DE CHIMLE EN COLABORATORY

3.1 Configuración de Google Generative AI para acceso a la API

En la Fig. 4, se realiza los pasos necesarios para configurar el entorno de trabajo con la biblioteca **google-generativeai**, que permite interactuar con modelos generativos proporcionados por Google.

Instalación de la biblioteca: Se asegura de que la versión más reciente (≥ 0.8.3) de google-generativeai esté instalada.

Importación de la biblioteca: Se importa el módulo necesario (google.generativeai) para trabajar con los modelos generativos.

Configuración de la API Key: Se configura el acceso a los servicios de Google Generative AI mediante una clave API (**GOOGLE_API_KEY**) proporcionada al usuario.

```
%pip install -q -U 'google-generativeai>=0.8.3'
import google.generativeai as genai
GOOGLE_API_KEY = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
genai.configure(api_key=GOOGLE_API_KEY)
```

Fig. 4 Configurar Google Generative IA

3.2 Creación y Configuración de un Esquema de la Base de Datos en SQLite

En la Fig. 5, se crean las tablas que usará CHIMLE. Primero, se carga la extensión %sql para habilitar la ejecución de comandos SQL dentro del notebook y se configura una base de datos SQLite llamada sample.db. Luego se define y crea un esquema de base de datos compuesto por las siguientes tablas relacionadas:

1. **Estudiantes:** Contiene información básica sobre los estudiantes, como nombre, apellido, DNI y correo.
2. **Docentes:** Almacena datos de los docentes, incluyendo su especialidad.

3. **Cursos:** Contiene información sobre los cursos ofrecidos, asociados a un docente.
4. **Inscripciones:** Registra la relación entre estudiantes y cursos, asegurando que no haya duplicados.
5. **Notas:** Almacena las calificaciones de los estudiantes en diferentes evaluaciones dentro de un curso.

También se establece claves primarias (PK) y foráneas (FK) para garantizar la integridad referencial y define restricciones como la unicidad para evitar datos duplicados en ciertos campos críticos.

```
%load_ext sql
%sql sqlite:///sample.db
%%sql
-- Tabla: Estudiantes
CREATE TABLE Estudiantes (
    id_estudiante INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    dni VARCHAR(15) UNIQUE NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL,
    fecha_registro DATE DEFAULT CURRENT_DATE
);
-- Tabla: Docentes
CREATE TABLE Docentes (
    id_docente INT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    dni VARCHAR(15) UNIQUE NOT NULL,
    correo VARCHAR(100) UNIQUE NOT NULL,
    especialidad VARCHAR(100)
);
-- Tabla: Cursos
CREATE TABLE Cursos (
    id_curso INT PRIMARY KEY,
    nombre_curso VARCHAR(100) NOT NULL,
    codigo_curso VARCHAR(10) UNIQUE NOT NULL,
    id_docente INT NOT NULL,
    creditos INT NOT NULL,
    semestre VARCHAR(10) NOT NULL,
    FOREIGN KEY (id_docente) REFERENCES Docentes(id_docente)
);
```

Fig. 5 Configurar base de datos

3.3 Llenado de las tablas: Estudiantes, Docentes, Cursos, Inscripciones y Notas.

Se llenaron con datos de prueba, en total 100 alumnos, 20 docentes, 10 cursos, 97 inscripciones y 100 notas.

3.4 Creación de las herramientas sql

Se definen en la base de dato de prueba, las funciones: list_tables ver Fig. 6, describe_table ver Fig. 7 y execute_query ver Fig. 8.

```
def list_tables() -> list[str]:
    """Recuperar los nombres de todas las tablas en la base de datos."""
    # Se imprime cuando la función es llamada.
    print(' - DB CALL: list_tables')

    cursor = db_conn.cursor()

    # Obtener los nombres de las tablas.
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")

    tables = cursor.fetchall()
    return [t[0] for t in tables]
```

Fig. 6 Función list_tables

```
def describe_table(table_name: str) -> list[tuple[str, str]]:
    """Consultar el esquema de la tabla.
    Devuelve:
          Una lista de columnas, donde cada entrada es una tupla con (columna, tipo).
    """
    print(' - DB CALL: describe_table')

    cursor = db_conn.cursor()

    cursor.execute(f"PRAGMA table_info({table_name});")

    schema = cursor.fetchall()
    # [column index, column name, column type, ...]
    return [(col[1], col[2]) for col in schema]
```

Fig. 7 Función describe_table

```
def execute_query(sql: str) -> list[list[str]]:
    """Ejecuta una sentencia SELECT , retornando los resultados."""
    print(' - DB CALL: execute_query')

    cursor = db_conn.cursor()

    cursor.execute(sql)
    return cursor.fetchall()

execute_query("select * from Estudiantes")
```

Fig. 8 Función execute_query

3.5 Configuración de un Chatbot Generativo para Consultas SQL Universitarias

Se configura un chatbot, ver Fig. 9, basado en un modelo generativo que interactúa con una base de datos SQL para responder preguntas de los usuarios. Los pasos principales son:

Definición de herramientas SQL (db_tools): Incluye funciones para listar tablas (list_tables), describir el esquema de las tablas (describe_table) y ejecutar consultas SQL (execute_query).

Instrucciones del sistema: Configura el comportamiento del chatbot mediante instrucciones para convertir preguntas de los usuarios en consultas SQL utilizando las herramientas definidas. El flujo incluye:

- Identificar tablas disponibles.
- Comprender la estructura de las tablas.
- Generar y ejecutar consultas SQL para obtener los datos necesarios.

Creación del modelo generativo:

- Utiliza el modelo gemini-1.5-flash-latest.
- Configura el modelo con las herramientas y las instrucciones.

Inicio de una sesión de chat: Permite al modelo realizar automáticamente llamadas a las funciones definidas (db_tools) según las necesidades detectadas en las consultas del usuario.

El chatbot creado puede responder preguntas de usuarios sobre la base de datos de la universidad, generando respuestas basadas en consultas SQL dinámicas.

```
# Herramientas sql definidas anteriormente.
db_tools = [list_tables, describe_table, execute_query]

instruction = """You are a helpful chatbot that can interact with an SQL database for a university.
You will take the users questions and turn them into SQL queries using the tools
available. Once you have the information you need, you will answer the user's question using
the data returned. Use list_tables to see what tables are present, describe_table to understand
the schema, and execute_query to issue an SQL SELECT query."""

model = genai.GenerativeModel(
    "models/gemini-1.5-flash-latest", tools=db_tools, system_instruction=instruction
)

# Se define una política de reintentos. El modelo podría realizar múltiples llamadas consecutivas automáticamente
# para una consulta compleja, lo que asegura que el cliente reintente si se alcanzan los límites de cuota.
from google.api_core import retry

retry_policy = {"retry": retry.Retry(predicate=retry.if_transient_error)}

# Inicia un chat con la llamada automática de funciones habilitada.
chat = model.start_chat(enable_automatic_function_calling=True)
```

Fig. 9 Configuración del Chatbot

IV. RESULTADOS Y PRUEBAS DE CHIMLE

Se realizaron pruebas con respecto a las notas, alumnos, docentes y cursos de la base de datos. Los resultados de las pruebas ejecutadas en CHIMLET, se muestran en la Tabla I.

Se realizaron 16 pruebas, con 13 pruebas con las respuestas correctas y 3 pruebas con respuestas incorrectas.

El análisis de las pruebas brinda resultados sobre la capacidad actual del prototipo:

Fortalezas en Lógica Matemática y Relacional

El modelo demostró una competencia avanzada en la construcción de sentencias SQL complejas.

En la Prueba 13, logró ejecutar operaciones de agregación (AVG), agrupación (GROUP BY) y ordenamiento (ORDER BY ASC LIMIT 3) para identificar correctamente los cursos con menor rendimiento ("Estadística y Probabilidades" con 12.47).

En la Prueba 14, el modelo navegó exitosamente la integridad referencial completa del sistema, uniendo cuatro tablas (Estudiantes →Inscripciones → Secciones → Cursos) para filtrar alumnos basándose en un atributo numérico del curso (créditos).

Debilidades en Semántica y Recuperación de Esquema

Se detectaron fallos críticos que no están relacionados con la capacidad de codificar SQL, sino con la interpretación semántica y la ventana de contexto:

Persistencia de la Ambigüedad (Prueba 12): A pesar de recibir una instrucción explícita ("Si hay varios, lístalos todos"), el modelo fusionó las identidades de múltiples docentes apellidados "Lee" en una sola entidad ficticia. Esto confirma que el modelo prioriza la recuperación de datos sobre la verificación de identidad, un comportamiento que debe corregirse mediante Prompt Engineering.

TABLA I

Resultados de las pruebas

Nro.	Pregunta (Resumida)	Tópico(s) Principal(es)	Resultado (Según Evaluación)
1	Mejor nota promedio en el curso SI707.	Estudiante, Curso, Nota (Promedio)	Correcta
2	(Seguimiento) ¿Quién es ella y qué notas tiene?	Estudiante (Contexto), Nota	Correcta
3	(Seguimiento) ¿Quién es el docente?	Docente (Contexto), Curso	Correcta pero imprecisa
4	Promedio de notas del Curso 3.	Curso, Nota (Promedio), Manejo de Nulos	Correcta (Manejo de nulos)
5	Listar promedio de notas para todos los cursos.	Curso (Todos), Nota (Promedio)	Correcta
6	Listar curso, docente y promedio de todos los cursos.	Curso (Todos), Docente, Nota (Promedio)	Correcta
7	Créditos totales de 'Rachel Bennett' en '2025-1'.	Estudiante, Periodo, Créditos	Correcta (Manejo de nulos/cero resultados)
8	¿Qué cursos enseña el profesor Lee?	Docente, Ambigüedad	Incorrecta (Incompleta)
9	¿Qué notas tiene el estudiante 'Elon Musk'?	Estudiante (Inexistente), Cero Resultados	Correcta (Manejo de cero resultados)
10	Correo de la profesora 'Debra Phillips'.	Docente, Correo	Correcta
11	(Seguimiento) ¿Qué asignaturas enseña ella?	Docente (Contexto), Curso	Correcta (Manejo de contexto)
12	Cursos que enseña el "Profesor Lee" (con solicitud explícita de desambiguación).	Ambigüedad, Identidad de Entidades	Incorrecta (Persistencia del fallo)
13	Los 3 cursos con menor promedio de notas (Ranking).	Agregación (AVG), Ordenamiento, Límites	Correcta
14	Alumnos en cursos de 6 créditos en el ciclo 2025-1.	Joins Múltiples (4 tablas), Filtrado Numérico	Correcta
15	Desglose de notas (Parcial/Final) de 'Tammy Harvey'.	Lectura de Esquema (Tabla Notas)	Incorrecta (Fallo de recuperación)
16	Promedio del curso inexistente 'Robótica...'.	Lógica Negativa, Manejo de Vacíos	Correcta

Fallo de Recuperación de Esquema (Prueba 15): El modelo sufrió una "alucinación de restricción". Afirmó que "no hay una tabla visible que contenga información detallada sobre evaluaciones", a pesar de que la tabla Notas está claramente definida en el esquema y vinculada a Inscripciones. Esto sugiere que, en consultas complejas, el modelo puede perder visibilidad de las tablas "hoja" (tablas al final de la jerarquía relacional).

IV. LIMITACIONES

La principal limitación actual es que la evidencia empírica disponible corresponde a una prueba piloto de 16 consultas tipos, es necesario realizar un mayor número de pruebas, considerando métricas estándares. Otra limitación es el uso de un entorno Colaboratory, adecuado para prototipado, pero es necesario un entorno institucional de prueba, donde se debe analizar también un análisis de seguridad y privacidad, Además, el desempeño puede variar según la versión del modelo LLM, parámetros de generación, estructura de la base, cantidad de registros de información y calidad de las instrucciones del sistema. Finalmente, debido a la naturaleza probabilística de los Modelos de Lenguaje Extensos (LLM), el sistema mantiene riesgo de alucinaciones, especialmente en consultas ambiguas, entidades homónimas o escenarios donde la recuperación del esquema de base de datos es incompleta.

CONCLUSIONES y DISCUSIÓN

A. CONCLUSIONES

Se desarrolló con éxito CHIMLE, un prototipo de chatbot implementado en Google Colaboratory que integra herramientas SQL y un modelo generativo para interactuar con las bases de datos académicas.

Las pruebas realizadas demuestran su eficacia en la interpretación y resolución de consultas complejas.

Este trabajo establece un procedimiento claro y replicable para el desarrollo de chatbots basados en LLM, contribuyendo a la gestión eficiente de información en instituciones educativas

B. DISCUSIÓN

En comparación con el modelo aplicado en la presente investigación, ambos enfoques comparten la utilización de modelos de lenguaje avanzados para facilitar la gestión y consulta de información académica. Sin embargo, mientras el artículo se centra en el uso de modelos preentrenados integrados con algoritmos específicos para extraer información de documentos en PDF y responder a consultas en un contexto de soporte universitario, nuestro desarrollo se enfoca en un prototipo implementado en Google Colaboratory que utiliza LLMs para gestionar y consultar información académica en una base de datos de alumnos. Además, en nuestro modelo, la gestión de la base de datos permite una interacción en tiempo real con datos estructurados y actualizados, complementando las capacidades de los modelos de lenguaje para ofrecer respuestas más precisas y contextualizadas a consultas académicas, con énfasis en la integración y gestión eficiente de datos propios de la institución educativa. [19]

Por otra parte, en comparación con enfoques recientes de construcción genérica de chatbots QA basados en LLM [20], nuestro proyecto busca ampliar este tipo de soluciones al integrar Modelos de Lenguaje Extensos (LLM) en la interacción con bases de datos institucionales, no solo para consultas frecuentes sino también para la generación de reportes personalizados y la automatización de tareas administrativas complejas. Mientras que el asistente virtual desarrollado en el artículo se limita a un conjunto predefinido de intenciones y respuestas, nuestro enfoque aprovecha la capacidad de los LLM para interpretar consultas en lenguaje natural con mayor flexibilidad y adaptabilidad. Esto abre la posibilidad de ofrecer un sistema escalable y aplicable a diversos contextos educativos, con mayor potencial de personalización en la experiencia del usuario y una integración más robusta con los sistemas de gestión académica existentes [20].

REFERENCES

.[1] H. Qiu, W. Hu, Z.-Y. Dou, and N. Peng, “VALOR-EVAL: Holistic Coverage and Faithfulness Evaluation of Large Vision-Language Models,” arXiv preprint, arXiv:2404.13874v4, 2024.

[2] M. Lassche, M. Goudbeek, and E. Krahmer, “Is our chatbot telling lies? Investigating answer correctness in human–chatbot conversations,” ACM Transactions on Computer-Human Interaction, vol. 31, no. 1, Art. no. 6, 2024, doi: 10.1145/3638538.

[3] K. S. Akuthota, A. S. Ganjam, K. K. Reddy et al., “Educational intelligent chatbot for Von Willebrand disease among Senegalese people,” in Proc. Int. Conf. Sustainable Expert Systems (ICSES), IEEE, 2024, doi: 10.1109/ICSES63445.2024.10763201

[4] F. Mangia, S. L. Risi, and S. Alivernini, “Discursively negotiating artificial intelligence: Social representations of chatbots in educational contexts,” Computers & Education: Artificial Intelligence, vol. 5, Art. no. 100159, 2024, doi: 10.1016/j.caeai.2024.100159.

[5] A. Fernández Domínguez, Large Language Models Inference and Deployment, B.Sc. thesis, Univ. Sevilla, Spain, 2023

[6] R. Jayakody and G. Dias, “Performance of recent large language models for a low-resourced language,” in Proc. Int. Conf. Asian Language Processing (IALP), IEEE, 2024, doi: 10.1109/IALP63756.2024.10661169.

[7] R. Lakshminarayanan, A. Shaik, and A. Balasundaram, “Automated speech therapy through personalized pronunciation correction using

reinforcement learning and large language models," Results in Engineering, vol. 25, Art. no. 103943, 2025, doi: 10.1016/j.rineng.2025.103943.
[8] C. Guan et al., "Leveraging large language models for peptide antibiotic design," Cell Reports Physical Science, 2024, doi: 10.1016/j.xcrp.2024.102359.
[9] I. Day et al., "TCMChat: A generative large language model for traditional Chinese medicine," Pharmacological Research, vol. 210, Art. no. 107530, 2024.
[10] J. Xiong, M. Wei et al., "Assessing the effectiveness of crawlers and large language models in detecting adversarial hidden link threats in meta computing," High-Confidence Computing, 2024, doi: 10.1016/j.hcc.2024.100292.
[11] F. Piccialli, "Federated and edge learning for large language models," Information Fusion, vol. 117, Art. no. 102840, 2025
[12] B. Ni, X. Cai, Z. Shen, Z. Meng, J. Zhao, Y. Cheng, and X. Gui, "Intelli-Dispatch-SQL: An LLM-based agent for reliable text-to-SQL in power dispatching," Energy and AI, vol. 22, Art. no. 100591, 2025, doi: 10.1016/j.egyai.2025.100591.
[13] W. Yang, L. Some, M. Bain, and B. Kang, "A comprehensive survey on integrating large language models with knowledge-based methods," Knowledge-Based Systems, vol. 318, Art. no. 113503, 2025, doi: 10.1016/j.knosys.2025.113503.
[14] P. Tocto, G. Huamaní et al., "Detecting whether an academic monograph was produced by an artificial intelligence system using machine learning," in Proc. 22nd LACCEI Int. Multi-Conf. Eng., Educ. Technol., 2024.
[15] P. Tocto, G. Huamaní, and P. Zuloaga, "Machine learning application in university management: Classification model for engineering student dropout in Peru," in Proc. 21st LACCEI Int. Multi-Conf., 2023.
[16] P. Tocto, G. Huamaní, and G. Villacorta, "Application of artificial intelligence in the management of a public university in Peru," in Proc. 20th LACCEI Int. Multi-Conf., 2022.
[17] P. Tocto, G. Huamaní, and G. Zuloaga, "Construction of a neural network-based model to determine the duration of engineering studies in a public university in Peru," in Proc. 19th LACCEI Int. Multi-Conf., 2021.
[18] Google DeepMind, "Gemini models," Gemini API Documentation, 2025. [Online]. Available: https://ai.google.dev/gemini-api/docs/models/gemini
[19] J. B. Ilagan and J. R. Ilagan, "A prototype of a conversational virtual university support agent powered by a large language model," Procedia Computer Science, vol. 239, 2024, doi: 10.1016/j.procs.2024.06.278.
[20] S. Salim, M. A. Hasan, and A. A. Rahman, "A large language model-based question answering chatbot builder for domain-specific knowledge," Expert Systems with Applications, vol. 240, Art. no. 122379, 2024, doi: 10.1016/j.eswa.2024.122379.