SISTEMA DE INTERFAZ NATURAL TEXT-TO-SQL PARA LA CONSULTA DE BASES DE DATOS PÚBLICAS DEL SECTOR DE SALUD Y PROTECCIÓN SOCIAL UTILIZANDO PROCESAMIENTO DE LENGUAJE NATURAL

NICOLÁS ANDRÉS RAMÍREZ CALDERÓN
ALEJANDRA ORTIZ MEDINA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISCOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2026

SISTEMA DE INTERFAZ NATURAL TEXT-TO-SQL PARA LA CONSULTA DE BASES DE DATOS PÚBLICAS DEL SECTOR DE SALUD Y PROTECCIÓN SOCIAL UTILIZANDO PROCESAMIENTO DE LENGUAJE NATURAL

NICOLÁS ANDRÉS RAMÍREZ CALDERÓN
ALEJANDRA ORTIZ MEDINA

Trabajo de Grado para optar al título de
Ingeniero de Sistemas

Director:
PhD. Bernardo Andrés Benavides Arévalo

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISCOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2026

CONTENIDO

pág.

LISTA DE FIGURAS

pág.

[1] Ashish Vaswani et al. «Attention is all you need». En: *Advances in neural information processing systems* 30 (2017).

LISTA DE TABLAS

pág.

LISTA DE ANEXOS

RESUMEN

TÍTULO: SISTEMA DE INTERFAZ NATURAL PARA LA CONSULTA DE BASES DE DATOS PÚBLICAS DEL SECTOR SALUD Y PROTECCIÓN SOCIAL UTILIZANDO PROCESAMIENTO DE LENGUAJE NATURAL*

AUTORES: NICOLÁS ANDRÉS RAMÍREZ CALDERÓN, ALEJANDRA ORTIZ MEDINA**

PALABRAS CLAVE: Procesamiento de Lenguaje Natural, *Text-To-SQL*, Modelos de Lenguaje de Gran Escala, Datos Abiertos, Bases de Datos, Sector Salud.

DESCRIPCIÓN:

La disponibilidad de datos abiertos gubernamentales se ha consolidado como un elemento clave para fortalecer la transparencia, la participación ciudadana y el desarrollo tecnológico. En Colombia, el Portal de Datos Abiertos reúne miles de conjuntos de datos públicos de diferentes sectores, entre los cuales destacan aquellos relacionados con salud y protección social por su impacto directo en la ciudadanía. No obstante, las alternativas actuales de consulta, como interfaces web, APIs y protocolos especializados de acceso a datos, requieren conocimientos técnicos que representan una barrera significativa para usuarios sin experiencia en bases de datos o lenguajes de consulta estructurados. Con el propósito de reducir estas limitaciones, este trabajo presenta el desarrollo de un prototipo de software basado en el paradigma Text-To-SQL, orientado a transformar consultas en lenguaje natural en sentencias SQL ejecutables mediante Modelos de Lenguaje de Gran Escala (Large Language Models, LLM). La arquitectura propuesta integra una interfaz de usuario, un módulo de inferencia semántica y una base de datos diseñada para consultar los conjuntos de datos Código Único de Medicamentos Vigentes (CUMV), Precios de Medicamentos (PM) y Listado de medicamentos de venta libre (LMVL). La validación del prototipo se realizó desde dos enfoques complementarios: el primero evaluó la precisión del módulo semántico mediante métricas del estado del arte en sistemas Text-To-SQL, y el segundo analizó la carga cognitiva y estructural asociada a distintos métodos de consulta sobre bases de datos públicas. Los resultados evidencian que la solución desarrollada facilita el acceso a los datos abiertos y constituye una alternativa reproducible y adaptable a otros dominios de información.

*Trabajo de grado

**Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: PhD. Bernanrdo Andrés Benavides Arévalo.

ABSTRACT

TITLE: NATURAL LANGUAGE INTERFACE SYSTEM FOR QUERYING PUBLIC DATABASES IN THE HEALTH AND SOCIAL PROTECTION SECTOR USING NATURAL LANGUAGE PROCESSING[*]

AUTHORS: NICOLÁS ANDRÉS RAMÍREZ CALDERÓN, ALEJANDRA ORTIZ MEDINA[**]

KEYWORDS: Natural Language Processing, *Text-To-SQL*, Large Language Models, Open Data, Databases, Health Sector

DESCRIPTION:

The availability of open government data has become a key element in strengthening transparency, citizen participation, and technological development. In Colombia, the Open Data Portal brings together thousands of public datasets from different sectors, among which those related to health and social protection stand out due to their direct impact on citizens. However, current access methods, such as web interfaces, APIs, and specialized data access protocols, require technical knowledge that represents a significant barrier for users without experience in databases or structured query languages. With the aim of reducing these limitations, this work presents the development of a software prototype based on the Text-to-SQL paradigm, designed to transform natural language queries into executable SQL statements using Large Language Models (LLMs). The proposed architecture integrates a user interface, a semantic inference module, and a database designed to query the Unique Code of Current Medications (CUMV), Drug Prices (PM), and Over-the-Counter Medicines List (LMVL) datasets. The prototype was validated from two complementary perspectives: the first evaluated the semantic module's accuracy using state-of-the-art metrics in Text-to-SQL systems, and the second analyzed the cognitive and structural load associated with different query methods over public databases. The results show that the developed solution facilitates access to open data and constitutes a reproducible and adaptable alternative for other information domains.

[*]Bachelor Thesis

[**]Faculty of Physical-Mechanical Engineering. School of Computer Science. Advisor: PhD. Bernardo Andrés Benavides Arévalo.

INTRODUCCIÓN

La democratización del acceso a la información pública se ha consolidado como un pilar fundamental para la transparencia gubernamental y la innovación tecnológica a nivel global. En este ámbito, Colombia se ha posicionado como un referente regional, poniendo a disposición de la ciudadanía el Portal de Datos Abiertos Colombia[2]. Este portal y la ley 1341 de 2009 y sus modificaciones le han permitido a Colombia tener un puntaje de $61{,}22$ según el *Global Data Barometer*, una iniciativa que mide cómo los países gestionan, gobiernan y utilizan los datos públicos[3]. A corte de 11 de febrero de 2025, el Ministerio de Tecnologías de la Información y las Telecomunicaciones (MinTIC) reportó que este portal registra un promedio de 200.000 búsquedas mensuales y 120.000 descargas de activos cada mes, lo que refleja una alta demanda por información estructurada[4].

Actualmente, el portal nacional de datos aloja más de 9.750 *datasets* públicos y 14.943 *datasets* privados, estos últimos para gestión interna de las entidades gubernamentales. Dentro de este ecosistema la categoría de Salud y Protección Social destaca como una de las de mayor interés ciudadano ya que cuenta con 386 solicitudes de apertura de datos desde 2016[5].

Un conjunto destacado dentro de esta categoría es el *Código Único de Medicamentos Vigentes* (CUMV), publicado por el INVIMA; cuenta con más de 1,4 millones de vistas, convirtiéndose en uno de los más consultados[6]. Este conjunto reúne información de medicamentos vigentes a nivel nacional, incluyendo registro sanitario, estado de vigencia y código único de medicamento identificado como CUM, así como detalles de presentación: principio activo, concentración, forma farmacéutica y

[2]Ministerio de Tecnologías de la Información y las Comunicaciones. *Datos Abiertos Colombia*. 2026. URL: https://www.datos.gov.co/.

[3]Ministerio de Tecnologías de la Información y las Comunicaciones. *Colombia alcanza el quinto lugar mundial en apertura de datos según el Global Data Barometer 2025*. 2025. URL: https://acortar.link/g3jRpv (visitado 27-08-2025).

[4]Ministerio de Tecnologías de la Información y las Comunicaciones. Dirección de Gobierno Digital. *Estudio Previo General. Formato: GCC-TIC-FM-001 V16*. Inf. téc. 2025. URL: https://www.mintic.gov.co/.

[5]Datos Abiertos Colombia. *Reportes Portal Nacional Datos Abiertos*. Datos Abiertos Colombia. 2025. URL: https://www.datos.gov.co/stories/s/Reportes-Portal-Nacional-Datos-Abiertos/pvyw-9yqs (visitado 10-09-2025).

[6]Datos Abiertos Colombia. *La Plataforma De Datos Abiertos Del Gobierno Colombiano*. 2025. URL: https://n9.cl/8lun4 (visitado 27-08-2025).

vía de administración, entre otros. Este conjunto de datos es altamente relevante, pues sirve como fuente autorizada para identificar los medicamentos vigentes en Colombia. También se ha utilizado en trabajos como el de Cuvides et al.[7], como parte de su investigación sobre el impacto económico del perfil epidemiológico sobre listado de medicamentos.

Actualmente la infraestructura tecnológica que utiliza el portal de Datos Abiertos Colombia opera bajo el modelo *software as a Service* (SaaS) proporcionado por Socrata, una empresa que se ha consolidado como un estándar tecnológico adoptado por múltiples países para la gestión de los datos abiertos[8]. Esta plataforma ofrece diferentes alternativas de consulta como la interfaz nativa de la página web, el protocolo *OData* y una Interfaz de Programación de Aplicaciones (API). Estas formas de acceso no son completamente intuitivas y pueden requerir conocimientos técnicos. Estas barreras técnicas constituyen un déficit de habilidades para el aprovechamiento de los datos, especialmente para usuarios inexpertos en conceptos de bases de datos[9].

Ante esta problemática, el paradigma de traducción *Text-To-SQL* emerge como una solución disruptiva. Este enfoque utiliza modelos de *Natural Language Processing* (NLP) para actuar como una capa de abstracción entre el lenguaje humano y los esquemas relacionales rígidos[10]. Al implementar interfaces conversacionales capaces de interpretar la intención semántica del usuario y transformarla en sentencias `SQL` ejecutables, se eliminan los requisitos de conocimiento técnico especializado, permitiendo una exploración de datos intuitiva y eficiente.

Con el fin de mitigar la complejidad de interacción con bases de datos, este trabajo presenta la implementación de un prototipo de *software* que integre una interfaz

[7] Hillary Cuvides López y Abimael Guerrero Vega. *Evaluación del impacto económico del perfil epidemiológico en el listado básico de medicamentos en institución hospitalaria Universidad del Norte*. Trabajo de grado. Universidad del Atlántico, 2023. URL: https://repositorio.uniatlantico.edu.co/server/api/core/bitstreams/5c9124c5-f3dc-4bef-874e-058afff2737b/content.

[8] Ministerio de Tecnologías de la Información y las Comunicaciones. Dirección de Gobierno Digital, *Estudio Previo General. Formato: GCC-TIC-FM-001 V16*.

[9] Norbert Lichtenauer et al. «A Scoping Review on Analysis of the Barriers and Support Factors of Open Data». En: *Information* 15.1 (2024). DOI: 10.3390/info15010005. URL: https://www.mdpi.com/2078-2489/15/1/5.

[10] Eduardo Nascimento et al. «Text-to-SQL Meets the Real-World». En: *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*. 2024, págs. 61-72. DOI: 10.5220/0012555200003690.

de usuario, un módulo de inferencia semántica basado en Modelos de Lenguaje de Gran Escala (*Large Language Models*, LLM) y una base de datos. El prototipo permite realizar consultas en lenguaje sobre los *datasets*: Código Único de Medicamentos Vigentes (CUMV), Clicsalud - Termómetro de Precios de Medicamentos (LVL) y Listado de medicamentos de venta libre (PM). En la validación del prototipo se utilizan dos enfoques; el primero gira en torno al módulo semántico validando su precisión con métricas del estado del arte en cuanto a paradigma *Text-To-SQL*; por otro lado, el segundo consiste en un enfoque relacionado con el análisis y la cuantificación de la carga cognitiva y estructural a la que se enfrenta el usuario al consultar las bases de datos a través de los diferentes métodos. Finalmente, se muestra cómo la arquitectura desarrollada en este trabajo permite su adaptación a diversos conjuntos de datos consolidándose como una alternativa reproducible y entendible.

1. OBJETIVOS

OBJETIVO GENERAL

Desarrollar un prototipo de *software* que posibilite consultas semánticas sobre conjuntos de datos abiertos del sector salud, con énfasis en el Código Único de Medicamentos Vigentes, lo cual el acceso a la información mediante técnicas de procesamiento de lenguaje natural.

OBJETIVOS ESPECÍFICOS

1. Implementar una base de datos relacional SQL para el conjunto de datos abierto del sector salud correspondiente al Código Único de Medicamentos Vigentes (CUMV), garantizando disponibilidad para consultas posteriores.
2. Crear un modelo de procesamiento de lenguaje natural (NLP) que transforme expresiones semánticas ingresadas por el usuario en peticiones SQL acordes a los atributos y relaciones del esquema de la base de datos.
3. Desarrollar un prototipo de *software* basado en una conexión entre el modelo NLP y la base de datos SQL, para la ejecución automática de consultas semánticas sobre la base de datos implementada.
4. Evaluar mediante métricas de complejidad los dos escenarios de interacción: (a) consultas directas en SQL sin intervención del bloque NLP y (b) consultas semánticas realizadas a través del bloque NLP, cuantificando el beneficio añadido del enfoque semántico para usuarios finales.

2. MARCO DE REFERENCIA

2.1 LENGUAJE DE CONSULTA ESTRUCTURADO (SQL)

El lenguaje de consultas *Structured Query Language* (SQL) es el estándar para la administración de bases de datos relacionales, cuyo propósito es extraer y recuperar la información sin modificar los registros subyacentes. Para garantizar que esta recuperación de datos sea exacta por el motor de base de datos, las instrucciones deben carecer de cualquier interpretación subjetiva. Por consiguiente, la sintaxis de SQL obedece a una estructura lógica estricta con nula tolerancia a la ambigüedad. Una consulta SQL típica se debe formular bajo parámetros predefinidos que establecen el origen, las condiciones de filtrado y el agrupamiento de los datos. Las cláusulas que rigen este tipo de operaciones son:

- ***SELECT*:** Define las columnas especificas que se desean recuperar.
- ***FROM*:** Especifica la tabla de donde se va a extraer la información.
- ***WHERE*:** Permite restringir el conjunto de resultados a partir de reglas lógicas.
- ***JOIN*:** Este operador permite integrar registros de múltiples tablas.
- ***GROUP BY* y Funciones de agregación:** Se utiliza para calcular métricas sobre subconjuntos de datos agrupados por dimensiones específicas. Algunas funciones son: *COUNT*, que sirve para contar registros; *SUM*, que suma todos los registros bajo parámetros de agrupación; y *AVG*, que calcula el promedio de registros definidos.

La rigidez de estas cláusulas es la principal barrera técnica para poder extraer información de bases de datos. Un error tipográfico o una asociación incorrecta, deriva en la ejecución incorrecta de dichas consultas.

2.2 PROCESAMIENTO DE LENGUAJE NATURAL (NLP)

El procesamiento de lenguaje natural (NLP) es una rama de las ciencias de la computación, la lingüística y la inteligencia artificial que estudia las interacciones entre los computadores y el lenguaje humano. Es un área con aplicaciones variadas, tales como la traducción automática, el reconocimiento y comprensión del lenguaje y la clasificación de texto entre otras[11].

En el estado del arte de la traducción de lenguaje natural a consultas SQL (*Text-To-SQL*), predomina el uso de los LLM. Gracias a su entrenamiento previo con corpus masivos de texto y código, pueden abordar de manera efectiva los retos clásicos del NLP, tales como la ambigüedad semántica, la naturaleza no estructurada del lenguaje y la fuerte dependencia del contexto.

2.3 ARQUITECTURA TRANSFORMER

La traducción de lenguaje natural a SQL, tiene un referente histórico en el modelo Seq2SQL[12], el cual demostró la viabilidad de generar consultas utilizando Redes Neuronales Recurrentes (RNN)[13]. Sin embargo la estructura secuencial de las RNN limitaba la retención del contexto en esquema relacionales complejos. Este desafío fue superado gracias a la arquitectura *Transformer*[14] la cual prescinde de la concurrencia y utiliza mecanismos de atención que analizan de forma simultánea toda la secuencia de entrada. Esta arquitectura se describe en detalle en la siguiente sección.

Como se ilustra en la Figura 1 el *transformer* se compone de una estructura *encoder-decoder*. El *encoder* recibe los *embeddings* (representaciones vectoriales de los to-

[11] Alberto García Serrano. *Inteligencia Artificial. Fundamentos, práctica y aplicaciones*. Rc Libros, 2012.

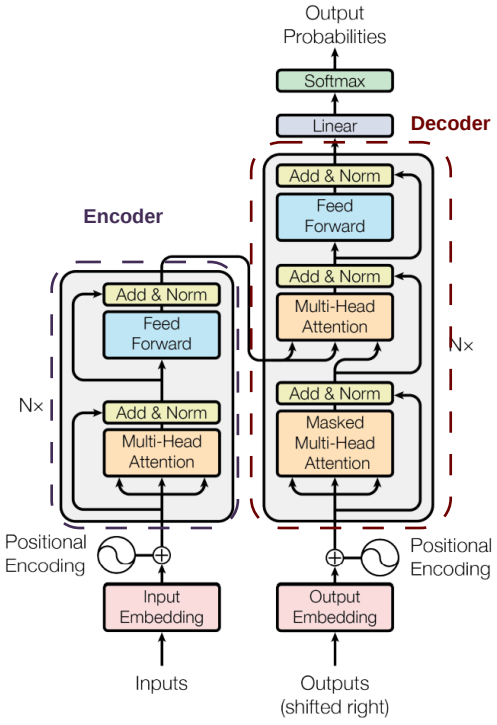
[12] Victor Zhong, Caiming Xiong y Richard Socher. *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning*. 2017. arXiv: 1709.00103 [cs.CL]. URL: https://arxiv.org/abs/1709.00103.

[13] Carlos Arana. *Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales*. Inf. téc. Serie Documentos de Trabajo, 2021.

[14] Ashish Vaswani et al. «Attention is all you need». En: *Advances in neural information processing systems* 30 (2017).

kens) enriquecidos con codificación posicional, los procesa a través de capas de autoatención multi-cabeza y redes *feed-forward* para garantizar una representación contextualizada de toda la consulta. Por su parte, el *decoder* utiliza esa representación para predecir la sentencia SQL de salida, empleando atención enmascarada para evitar que el modelo acceda a posiciones futuras durante la generación.

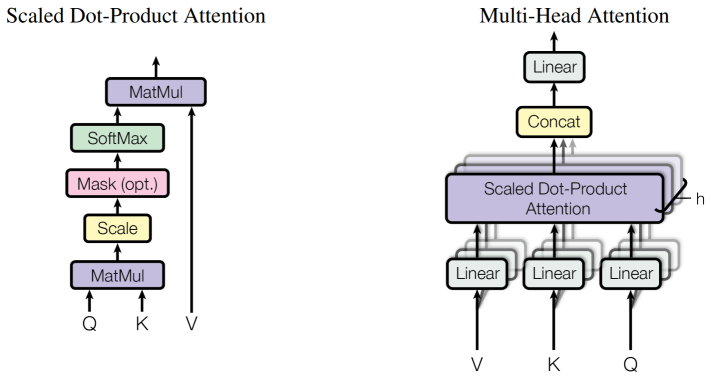
Figura 1. Arquitectura original del Transformer[15]. El bloque *encoder* procesa los *embeddings* mediante auto-atención multi-cabeza para generar una memoria contextual. El *decoder* utiliza esta representación, combinada con autoatención enmascarada, para generar la secuencia de salida de manera autorregresiva.



El núcleo de esta arquitectura son los mecanismos de atención que se representan en la Figura 2, diseñados para estimar la relevancia de cada token respecto a los demás sin importar su distancia a lo largo de la sentencia. En la autoatención cada token se proyecta en tres vectores: *Query* (Q), *Key* (K) y *Value* (V). Los pesos de atención son obtenidos mediante un producto punto escalado entre Q y K, el cual se normaliza con una función *Softmax*. Estos pesos ponderan los valores (V) para

capturar dependencias semánticas globales[16]. Simultáneamente la variante multi-cabeza ejecuta este proceso en paralelo, permitiendo al modelo capturar múltiples patrones sintácticos, una característica vital para comprender intenciones complejas en consultas en bases de datos.

Figura 2. (Izquierda) Scaled-Dot-Product Attention (Derecha) Multi-Head Attention.



2.4 MODELOS DE LENGUAJE DE GRAN ESCALA

Un Modelos de Lenguaje de Gran Escala (LLM) es una red neuronal profunda compuesta por millones de parámetros, diseñado para comprender, extraer y generar lenguaje natural. Estos modelos se fundamentan en la arquitectura *transformer*[17], la cual les otorga la capacidad de analizar secuencias de texto y abstraer patrones semánticos complejos mediante el uso intensivo de mecanismos de atención.

En el contexto de este trabajo, los modelos implementados corresponden a la variante de arquitectura *decoder-only* (compuesta únicamente por el bloque decodificador). Estos operan bajo un paradigma autorregresivo, es decir generan la secuencia de salida al predecir iterativamente el siguiente *token* condicionados solo por el contexto de la información previa.

[16] André Augusto Suave. *Inteligência Artificial*. ProQuest Ebook Central. Freitas Bastos, 2024. URL: https://ebookcentral.proquest.com/lib/bibliouis-ebooks/detail.action?docID=31498447; Vaswani et al., «Attention is all you need».

[17] Vaswani et al., «Attention is all you need».

2.5 PARADIGMA DE TRADUCCIÓN TEXT-TO-SQL

La traducción de texto a SQL *Text-To-SQL* tiene como objetivo transformar preguntas formuladas en lenguaje natural en consultas estructuradas y ejecutables sobre bases de datos[18]. En la actualidad este paradigma se aborda desde dos enfoques metodológicos gracias a las capacidades de los modelos de lenguaje[19].

2.5.1 Enfoque de generación directa.

El enfoque de generación directa o *End-To-END*, integra un LLM que recibe la pregunta del usuario y el esquema de la base de datos como entrada y devuelve la cadena de texto con la sintaxis SQL final. Aunque es directo, este enfoque es altamente propenso a errores.

Bajo este método los LLM sufren de alucinaciones, un término que se refiere a que un LLM infiere o genera contenido que no siempre es correcto. En el caso del paradigma *Text-To-SQL* estas alucinaciones son un problema pues los modelos podrían generar sentencias lógicamente incorrectas que fallan durante su ejecución[20].

2.5.2 Enfoque modular.

El enfoque modular surge como una alternativa robusta para mitigar el problema de las alucinaciones, integrando un enfoque modular o *Descomposed Text-To-SQL*, el cual ha demostrado superar significativamente a los enfoques de generación directa[21].

Bajo este modelo se desacopla la comprensión semántica de la generación de código. El LLM es utilizado exclusivamente para extraer la intención de la consulta e identificar parámetros lógicos como; filtros, métricas, orden y límite. Posteriormente esta información pasa por un proceso de normalización semántica guiada por el esquema estricto de la base de datos y se integra en un módulo algorítmico que

[18] Bowen Qin et al. «A survey on text-to-sql parsing: Concepts, methods, and future directions». En: *arXiv preprint arXiv:2208.13629* (2022).

[19] Xinyu Liu et al. «A Survey of Text-to-SQL in the Era of LLMs: Where are we, and where are we going?» En: *IEEE Transactions on Knowledge and Data Engineering* (2025).

[20] Wenbo Xu et al. «TS-SQL: Test-driven Self-refinement for Text-to-SQL». En: *Findings of the Association for Computational Linguistics: EMNLP 2025*. 2025.

[21] Mohammadreza Pourreza y Davood Rafiei. «Din-sql: Decomposed in-context learning of text-to-sql with self-correction». En: *Advances in neural information processing systems* 36 (2023), págs. 36339-36348.

ensambla la consulta SQL, garantizando así una alta precisión semántica.

En este trabajo se implementa el enfoque modular en dos fases. La primera fase de naturaleza estocástica que emplea un LLM local como motor de procesamiento de lenguaje natural para extraer la intención de la consulta del usuario. La segunda fase es de caracter determinista, toma la estructura lógica extraída por el LLM para ejecutar una serie de rutinas programáticas de validación y posterior ensamblaje. Este proceso mitiga significativamente los errores de alucinación del LLM y garantiza que las consultas resultantes cumplan con los requerimientos rígidos del gestor de base de datos.

2.6 MÉTRICAS DE DESEMPEÑO TEXT-TO-SQL

Para realizar la evaluación del rendimiento de los modelos *Text-To-SQL* se han propuesto diferentes métricas el estado del arte. A continuación se presentarán las más relevantes para el presente trabajo de investigación.

2.6.1 Exactitud global (EX Score).

El EX Score es una métrica que calcula el error a nivel de resultado de la consulta SQL en lugar de a nivel de la propia sentencia SQL[22]. En esencia, esta métrica compara el resultado de ejecutar la sentencia SQL en la base de datos, en lugar de evaluar la propia sentencia por si sola. Formalmente esta definido como se expone en la ecuación (2.1).

$$EX = \frac{1}{N} \sum_{i=1}^{N} 1(R_{pred}^{(i)} = R_{gold}^{(i)}) \tag{2.1}$$

Donde R_{pred} es el resultado que se genera por el SQL predicho por el modelo, R_{gold} es el resultado de referencia.

Al comparar R_{pred} y R_{gold} se obtiene un indicador binario que vale 1 si estos resultados coinciden y 0 si no lo hacen. Este proceso se repite sobre un conjunto de N preguntas de validación.

[22]Geunyeong Jeong et al. «Improving Text-to-SQL with a Hybrid Decoding Method.» En: *Entropy (Basel, Switzerland)* 25.3 (2023). DOI: 10.3390/e25030513.

2.6.2 Evaluación por componentes (F1-Score). Para la evaluación por componentes se usan métricas clásicas como *precision*, *recall* y *F1-Score*[23], para cada uno de los componentes de una sentencia SQL. Es decir el cálculo se realiza para cada una de las cláusulas *SELECT*, *WHERE* y *GROUP BY*.

El cálculo del *F1-Score* usa los siguientes conceptos:

- True Positive (TP): Número de elementos correctamente generados por el modelo en una cláusula
- False Positive (FP): Número de elementos innecesariamente generados por el modelo en una cláusula
- False Negative (FN): Número de elementos necesarios no generados por el modelo en una cláusula

Con base en estos elementos es posible calcular las métricas necesarias, como se evidencia en las ecuaciones (2.2), (2.3) y (2.4).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.4}$$

A pesar de que la evaluación independiente de las cláusulas permite un mayor nivel de detalle para la evaluación del modelo, algunos autores han propuesto formas de promediar el *F1 Score* de todas las cláusulas del sistema para obtener una métrica más simple de comparar con otros modelos del estado del arte[24]. Siguiendo esta

[23] Tao Yu et al. «Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task». En: *ArXiv* abs/1809.08887 (2018). URL: https://api.semanticscholar.org/CorpusID:52815560.

[24] George Katsogiannis-Meimarakis y Georgia Koutrika. «A survey on deep learning approaches for text-to-SQL». En: *The VLDB Journal* 32 (ene. de 2023). DOI: 10.1007/s00778-022-00776-8.

tendencia, este trabajo hace uso del concepto de *Weighted F1 Scores*, para el cuál se asignan pesos entre 0 y 1 a cada uno de los *F1 Scores* y se suman los resultados tal como se describe en la ecuación (2.5).

$$\text{WF1} = W_S F1_{SELECT} + W_W F1_{WHERE} + W_O F1_{GROUP\ BY} + W_A F1_{AGGREGATION} \quad (2.5)$$

Donde $0 \leq W_S, W_W, W_O \leq 1$ y $W_S + W_W + W_O = 1$.

Esta métrica permite evaluar el desempeño general de los modelos, dándole un peso a cada cláusula dependiendo del contexto del problema y de la base de datos con la que se esté trabajando.

2.7 COMPLEJIDAD ESTRUCTURAL Y CARGA COGNITIVA

En esa sección se escriben las métricas que permiten comparar el nivel de complejidad de acceso a la información contenida en las bases de datos, bajo diferentes modelos de interacción.

2.7.1 Complejidad de Yaqin. La *Métrica de Complejidad de Yaqin (YC)*[25] es un método matemático diseñado originalmente para evaluar modelos de procesos de negocio a partir de su estructura de control básico. Según Yaqin et al.[26], los modelos de interacción y procesos están compuestos por patrones específicos cuyas características definen su nivel de dificultad operativa. La métrica de Yaqin proporciona una evaluación sobre carga cognitiva del proceso, conceptualmente establece que la carga de un flujo de trabajo no depende del contenido de la actividad en si, sino de la arquitectura para la toma de decisiones. El modelo de *Yaqin* cuantifica la fricción operativa, evaluando de manera simultanea alteraciones menores en la arquitectura del flujo, el tipo de lógica de ramificación, la cantidad de bifurcaciones, los bucles y

[25] Muhammad Ainul Yaqin, Riyanarto Sarno y Siti Rochimah. «Measuring Scalable Business Process Model Complexity Based on Basic Control Structure». En: *International Journal of Intelligent Engineering and Systems* 13 (dic. de 2020), págs. 52-65. DOI: 10.22266/ijies2020.1231.06.

[26] Ibíd.

la profundidad promedio del proceso.

En la validación y análisis de interfaces de software, por ejemplo, la aplicación de este modelo permite traducir la forma de navegación de un usuario en un valor numérico unificado, demostrando matemáticamente que al anidar más decisiones se incrementa drásticamente el esfuerzo cognitivo requerido para alcanzar un objetivo. Esta métrica está dada por la Ecuación 2.6 y es completamente definida en el Anexo E.

$$YC = Ns + As + C_{AND} + C_{XOR} + C_{OR} + C_{bucle} + C_D \tag{2.6}$$

- **Número de nodos (Ns):** Este parámetro es la sumatoria de los elementos que representan inicio, fin, actividades y decisiones dentro del diagrama.
- **Número de arcos (As):** Representa la cantidad total de conexiones que determinan la secuencia de ejecución (aristas o flechas).
- **Complejidad de bifurcaciones paralelas (C_{AND}):** Este parámetro representa la cantidad de actividades que se ejecutan en paralelo.
- **Complejidad de bifurcaciones exclusivas (C_{XOR}):** Esta complejidad representa la sumatoria de los elementos de toma de decisiones, es decir, las rutas excluyentes de un solo camino presentes en el flujo.
- **Complejidad de bifurcaciones inclusivas (C_{OR}):** Evalúa la complejidad de decisiones no estructuradas en las que el usuario puede tomar uno, varios o todos los caminos al mismo tiempo.
- **Complejidad de ciclos (C_{bucle}):** Mide la complejidad en términos de la repetición de una serie de pasos, es decir cuando el proceso retrocede a una actividad anterior. Este parámetro es la relación que existe entre las actividades involucradas en el ciclo frente al tamaño total de las actividades del flujo.
- **Complejidad de profundidad (C_D):** Mide el desgaste cognitivo acumulado por la longitud del proceso al usuario a recordar el contexto inicial. La Ecuación 2.7 calcula el promedio de profundidad (D_{avg}).

$$Davg = \frac{\sum_{j}^{Acs} \cdot D_j}{Acs} \tag{2.7}$$

En donde Acs es el número total de actividades, j es el indice iterativo de cada actividad y D_j es la profundidad de esa actividad (el número de arcos transitados desde el nodo inicial). La Ecuación 2.8 describe la complejidad de profundidad.

$$C_D = CW_{C_D} \cdot Davg \tag{2.8}$$

Cada parámetro de la *Métrica de Yaqin* representa un elemento específico del proceso, al cual se le añade un factor multiplicativo CW que representa una penalización por el costo cognitivo que tiene el usuario sobre ese elemento. A continuación se describen en detalle cada uno de los elementos de esta métrica.

Al aplicar la Ecuación 2.6 en escenarios reales, es probable que algunos parámetros de control no estén presentes en el diagrama evaluado, por lo cual su valor individual será cero. Para establecer los factores multiplicativos (CW) en este trabajo, se adoptaron los pesos de referencia establecidos por Yaqin et al.[27], los cuales penalizan severamente las decisiones y las rutas profundas. Estos valores se detallan en la Tabla 1.

Tabla 1. Pesos cognitivos aplicados como factor multiplicativo a los parámetro de la métrica de Yaqin.

Estructura / Parámetro	**Peso Cognitivo**
Secuencia (Ns, As)	1
Bifurcación Exclusiva (XOR)	3
Bifurcación (AND)	4
Bifurcación (OR)	7
Bucle o Ciclo	3
Profundidad de navegación	14

2.7.2 Complejidad Ciclomática de McCabe. La Complejidad Ciclomática es una métrica de ingeniería de software fundamentada en la teoría de grafos que cuantifica

[27] Yaqin, Sarno y Rochimah, «Measuring Scalable Business Process Model Complexity Based on Basic Control Structure».

la complejidad lógica de un sistema o proceso. Mide el número máximo de caminos de ejecución linealmente independientes dentro de un flujo de control. En el análisis de procesos, un valor más alto indica una mayor cantidad de bifurcaciones y puntos de decisión lógica, lo que incrementa la probabilidad de errores y exige un mayor esfuerzo mental para la comprensión de las rutas posibles por parte del usuario[28]. Esta métrica está dada por la Fórmula 2.9.

$$C_{cyc} = E - N + 2 \cdot P \tag{2.9}$$

En donde E representa el número de arcos„ N representa el número de nodos y P es el número de componentes conexos.

2.7.3 Complejidad de Procesos. La Complejidad del Proceso es una métrica operativa diseñada para cuantificar la carga cognitiva y el esfuerzo técnico requeridos para ejecutar una tarea dentro de un marco de trabajo. A diferencia de las métricas puramente topológicas, que evalúan la estructura del software, esta métrica se centra en la fricción interactiva desde la perspectiva del usuario. Se calcula sumando el nivel de dificultad inherente a cada paso operacional necesario para transitar desde el inicio de una tarea hasta su resolución final[29].

Matemáticamente, la complejidad del proceso (CH) se define mediante la Ecuación 2.10:

$$CH = \sum_{i=1}^{n} w_i \tag{2.10}$$

Donde n representa el número total de pasos operacionales requeridos para completar la tarea, y w_i es el peso cognitivo asignado al paso i.

Según la literatura, este peso cognitivo se categoriza en una escala ordinal, para este trabajo la escala aplicada fue de 0 a 3, dependiendo del grado de conocimiento

[28] Andrés Benavides et al. «Lithops-HPC: Extending the Serverless Paradigm to High-Performance Computing for Accessible Resource Management». En: *Available at SSRN 6174512* ().

[29] Ibíd.

técnico e intervención manual exigido:

Peso 0 (Automático): Ejecución completamente autónoma por parte del sistema informático, por ejemplo, tareas pasivas, como la visualización de resultados sin intervención del usuario.

Peso 1 (Básico): Interacciones estándar de interfaz de usuario, operaciones rutinarias, por ejemplo, el ingreso de instrucciones en lenguaje natural.

Peso 2 (Intermedio): Tareas que demandan configuración de parámetros técnicos, por ejemplo, manipulación estructurada de datos o gestión de variables de entorno.

Peso 3 (Avanzado): Actividades de alta carga cognitiva, por ejemplo que exigen el dominio de lenguajes de programación, sintaxis de consulta estructurada o configuración de protocolos de red.

3. MÉTODO PROPUESTO

En este capítulo se presenta el desarrollo de un prototipo de *software* para consultar bases de datos públicas del sector Salud y Protección Social en Colombia. El prototipo cuenta con la capacidad de recibir una pregunta en lenguaje natural, extraer su intención semántica utilizando un LLM y procesar esa intención con el fin de ensamblar una consulta SQL válida. Adicionalmente, también se muestra la forma en la que se aplicaron diferentes métricas de evaluación para validar todos los componentes del sistema propuesto y la complejidad de acceso a la información.

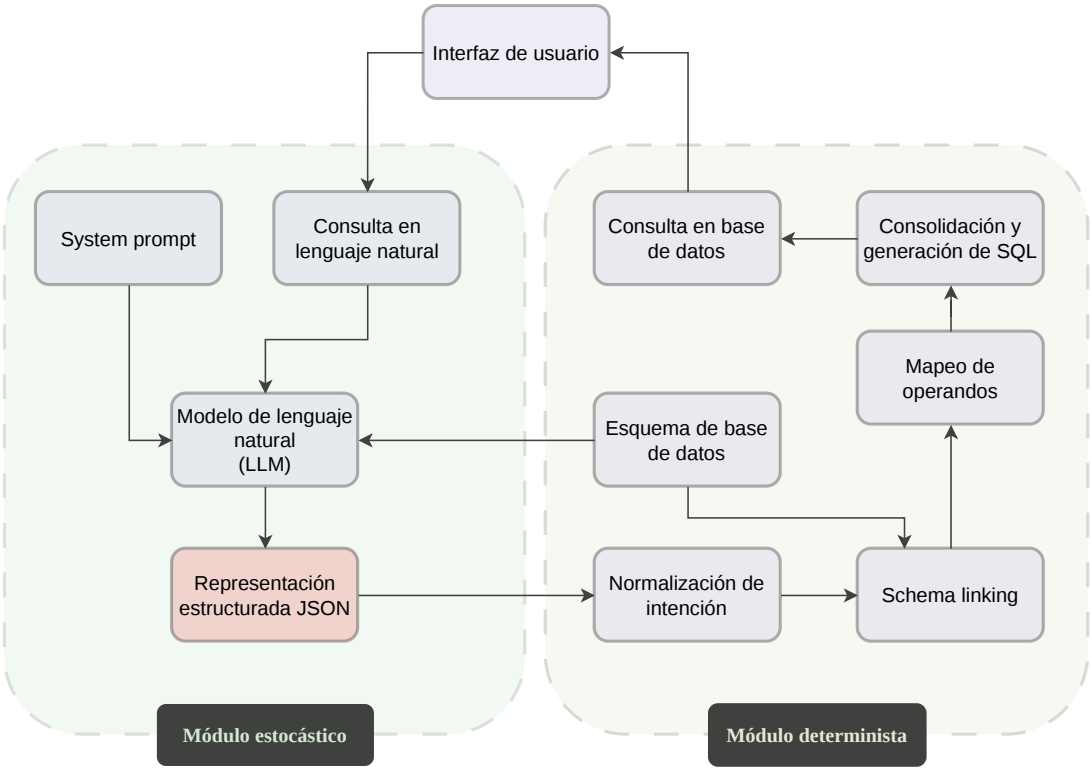
3.1 ARQUITECTURA GENERAL DEL SISTEMA TEXT-TO-SQL (PIPELINE)

La arquitectura diseñada para este trabajo se basó en un enfoque de dos módulos independientes (Fig. 3); (1) El *módulo estocástico* es el encargado de recibir la pregunta semántica del usuario y extraer su intención en un formato definido. (2) El módulo *módulo determinista* es aquel que se encarga del post-procesamiento de la intención, el envío de la consulta a la base de datos y la presentación del resultado al usuario. La estructura de estos modelos se detalla a continuación.

3.1.1 Módulo estocástico. La parte central de este módulo es el LLM encargado de procesar la consulta en lenguaje natural que el usuario ingrese al sistema. El modelo recibe la consulta y a partir de un *prompt* extrae entidades clave (tablas, columnas, condiciones, métricas y límites). El *Prompt* es un objeto estructurado en formato JSON, que contiene las reglas de salida y las instrucciones detalladas del formato de salida. Un ejemplo se muestra en el anexo B.

3.1.2 Módulo determinista. Este módulo es el encargado de realizar el pos-procesamiento del objeto JSON que que fue entregado por el módulo estocástico. En este módulo, primero se ejecuta una rutina de normalización, asegurando que los nombres de las columnas y los tipos de datos inferidos coincidan estrictamente con los esquemas de la base de datos real. Una vez validados estos parámetros, el algoritmo ensambla la cadena final de código SQL y esa consulta se envía al gestor de base de datos, el cual devuelve los registros solicitados para ser visualizados por el usuario en una

Figura 3. Arquitectura de sistema propuesto. Módulo estocástico (izq) en donde el LLM extrae la intención y la mapea en formato JSON. Módulo determinista (der) en el que un algoritmo post-procesa la intención y ensambla la consulta SQL para ser ejecutada en el gestor de base de datos enviando posteriormente el resultado al usuario.



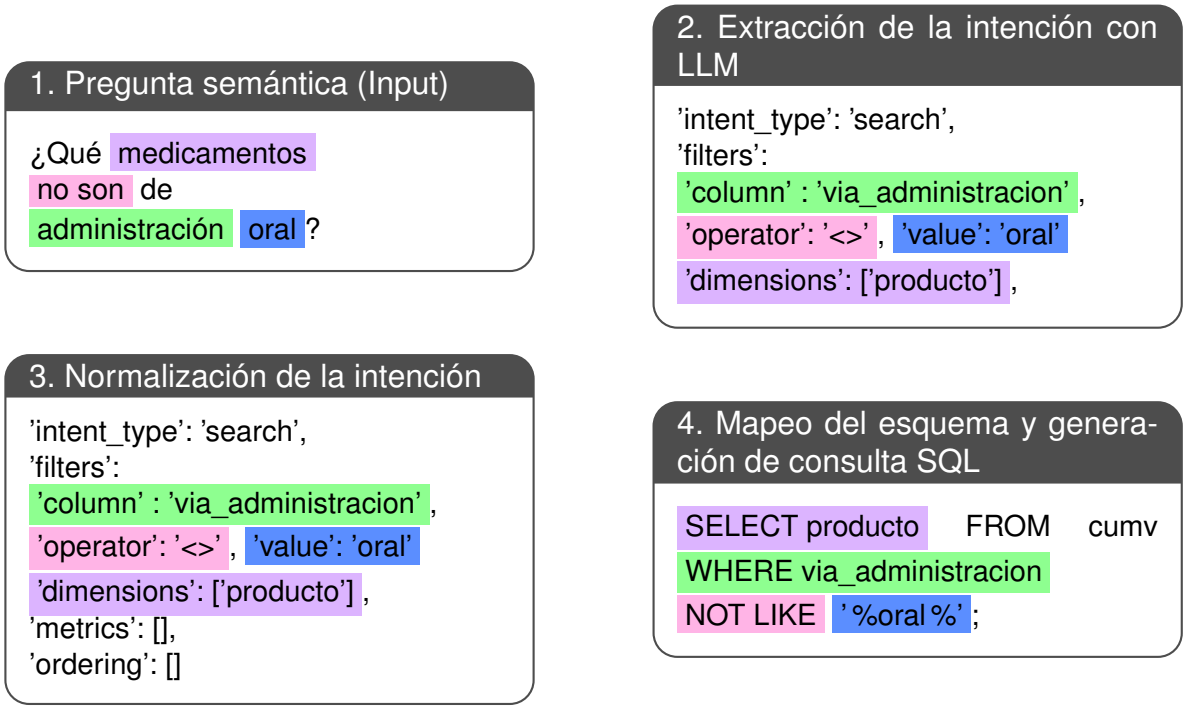
interfaz web.

3.2 FLUJO DE EJECUCIÓN DEL SISTEMA TEXT-TO-SQL

La figura 4 ilustra el funcionamiento completo del sistema. Este está conformado por cuatro etapas fundamentales: (1) la pregunta semántica inicial, es decir, el input del modelo, (2) la extracción de la intención realizada por el modelo LLM, (3) el resultado de la primera fase del post-procesamiento llamada normalización de la intención, y (4) la salida del modelo que integra un mapeo del esquema y posteriormente la

generación de la consulta SQL.

Figura 4. Flujo completo desde la pregunta semántica hasta la sentencia SQL final. Cada etapa se presenta en un bloque separado y las secciones correspondientes a cada cláusula se muestran con colores. La cláusula *SELECT* se denota con el color magenta, la cláusula *WHERE* está compuesta por: la columna que se denota con verde, el operador que se denota con rosa y el valor que se denota con el azul.



3.3 IMPLEMENTACIÓN DEL MODELO CON LA BASE DE DATOS

Para este trabajo se exploraron tres bases de datos los cuales fueron descargados del Portal de Datos Abiertos Colombia[30]. Estos conjuntos corresponden al sector de Salud y Protección Social y fueron seleccionados debido a su alta relevancia en el total de accesos por usuario. A continuación se listan los conjuntos de datos:

- Código Único de Medicamentos Vigentes (CUMV) [31]:

[30] Ministerio de Tecnologías de la Información y las Comunicaciones, *Datos Abiertos Colombia*.
[31] CUMV Datos Abiertos Colombia

Información de los códigos únicos de medicamentos para los registros sanitarios vigentes.

- Listado de medicamentos en venta libre (LMVL) [32]:

  Listado de los medicamentos que no requieren fórmula médica.

- Precios de medicamentos (PM) [33]:

  Precios equivalentes por tableta o cápsula de diferentes alternativas en el mercado para diferentes tipos medicamentos.

En general, se realizó un proceso que consistió en una limpieza de valores nulos, estandarizacion de atributos temporales y por último los datos pre-procesados se migraron a bases de datos relacionales sobre las que se realizaron pruebas de validación de integridad.

3.3.1 Preprocesamiento de datos. En esta fase se realizó un análisis, estandarización y limpieza de los conjuntos de datos con el objetivo de garantizar su integridad antes de ser migradas al motor de base de datos. Este proceso es fundamental para las siguientes fases como la creación y evaluación del modelo Text-To-SQL ya que se obtiene como resultado una base de datos sin registros errones que impidan el correcto funcionamiento del modelo.

Para esta tarea se utilizaron herramientas de *Python* como librería *Pandas* para obtener un resultado adecuado mediante pasos claros y reproducibles.El preprocesamiento general aplicado a las bases de datos se detalla a continuación:

- **Eliminación de elementos nulos:** Se realizó un análisis que permitió encontrar y eliminar registros duplicados, valores nulos y datos incorrectos. También se verificó la no existencia de columnas irrelevantes con datos consecutivos como los identificadores numéricos de los elementos en la tabla.
- **Estandarización de atributos temporales:** Los datos de tipo fecha son fundamentales en cada una de las bases de datos, pues son datos sobre los

[32] LMVL Datos Abiertos Colombia
[33] PM Datos Abiertos Colombia

cuales es posible diseñar diversas consultas, por lo tanto es fundamental realizar una validación de estos atributos. En una primera instancia se avanzó con un análisis de las estructuras de fecha presentes en las bases de datos. Posteriormente los valores de fecha identificados estandarizaron según *ISO 8601* (*YYYY-MM-DD*)[34] y por último se adaptaron los registros con valores de fecha incorrectos para mantener sus contenidos en otras columnas sin comprometer las futuras consultas temporales.

3.3.2 Construcción de base de datos SQL y validación de integridad. Una vez concluido el primer proceso de limpieza, los datos fueron migrados a un Sistema Gestor de Base de Datos Relacionales. El motor seleccionado fue *MariaDB*; esta decisión se fundamenta en su naturaleza de código abierto, lo cual mitiga el riesgo de dependencia tecnológica frente a dependencia privativas, garantizando al portabilidad y soberanía de los datos públicos.

Los valores limpios previamente almacenados en formato *CSV* se importaron a una base de datos mediante procedimientos en *Python* que incluyeron la creación de la tabla, las columnas con sus respectivos tipos y la importación fila por fila de cada uno de los registros.

Para validar la correcta estructura e integridad de los esquemas generados se creó un banco de preguntas formuladas en lenguaje natural emparejadas con sus respectivas consultas SQL de referencia (R_{gold}). Este banco se diseñó cubriendo diversos niveles de complejidad lógica (Tab. 2).

[34] International Organization for Standardization. *ISO 8601 — Date and time format*. ISO. 21 de feb. de 2017. URL: https://www.iso.org/iso-8601-date-and-time-format.html (visitado 30-03-2026).

Tabla 2. Clasificación lógica de consultas para la formulación de un banco de prueba para evaluar la integridad de los esquemas de las bases de datos.

Tipo de consulta	Cláusula / operador
Filtrado simple	WHERE
Filtrado multicondicional	AND, OR, NOT
Funciones de agregación	COUNT, MAX, MIN, AVG, SUM
Operaciones relacionales	>, <, =, !=
Búsqueda por rangos	BETWEEN
Estructuración y ordenamiento	ORDER BY (ASC, DESC)
Agrupamiento	GROUP BY
Filtrado de unicidad	DISTINCT

El objetivo del banco de preguntas semánticas es el de ser utilizadas en fases posteriores para entrenar, validar y evaluar a los diferentes modelos LLMs así como al sistema Text-To-SQL en general.

3.4 IMPLEMENTACIÓN DEL MÓDULO ESTOCÁSTICO

Como se explicó en la sección 3.1.1 módulo realiza la descomposición semántica de la consulta de usuario para transformarla en una estructura lógica, mitigando la ambigüedad del lenguaje natural[35]

3.4.1 Selección y despliegue del modelo de Lenguaje. Durante la selección del modelo de lenguaje que se utilizó en la arquitectura de este proyecto, se realizó una búsqueda orientada a identificar modelos de arquitectura abierta, que estuviesen entrenados para la generación de código y razonamiento lógico y que pudieran ser ejecutados de forma local. En este sentido se utilizó la herramienta *Ollama*, que funciona como una interfaz de código abierto que permite descargar y ejecutar modelos de forma local. Posterior a una primera búsqueda simple, se propusieron cuatro modelos candidatos iniciales con diferentes características (Tab. 3). Para la decisión final solo se tuvieron en consideración modelos con una cantidad baja de parámetros

[35]Qin et al., «A survey on text-to-sql parsing: Concepts, methods, and future directions».

para facilitar el posterior desarrollo de las pruebas iterativas y la experimentación necesaria sobre los mencionados LLMs requerida para esta investigación.

Tabla 3. Modelos de lenguaje evaluados para la extracción de la intención semántica. Todos son modelos abiertos.

Modelo	Familia	Parámetros (Billones)
Gemma 2	Google	2B
LLaMA 3.1	Meta	8B
Phi-3 Mini	Microsoft	3.8B
Qwen 2.5	Alibaba	7B

En la tarea de selección del modelo LLM motor del sistema Text-To-SQL, se llevó a cabo un proceso iterativo en el que se ajustaron manualmente los parámetros del modelo, así como también se realizaron modificaciones a la estructura del prompt ingresado por el sistema (input), con el fin de evaluar el mejor funcionamiento de cada uno. Estos resultados se describen y explican en el siguiente capítulo.

3.4.2 Construcción del diccionario de datos y Schema Linking. En aras de mejorar la capacidad el modelo en tareas de intepretación del lenguaje natural hacia la estructura técnica de una base de datos, se implementó una estrategia de *Schema Linking*, identificada como una fase crítica para el éxito de sistemas *Text-To-SQL*[36]. Esta fase comprendió la implementación de un diccionario semántico que mapea términos coloquiales o ambiguos hacia los identificadores reales de la base de datos (tablas, columnas o entidades).

Para garantizar la precisión del proceso de *Schema Linking*, se definieron los esquemas relacionales de los tres dominios de estudio. Las tablas 4, 5 y 6 detallan dichos esquemas relacionales.

[36]Pourreza y Rafiei, «Din-sql: Decomposed in-context learning of text-to-sql with self-correction».

Tabla 4. Esquema técnico normalizado de la base de datos CUMV.

Tipo de dato	Columnas (rename_map)
INTEGER	id, expediente, consecutivo_cum
STRING	producto, titular, registro_sanitario description_comercial, muestra_medica, unidad atc, description_atc, via_administracion concentracion, principio_activo, unidad_medida unidad_referencia, forma_farmaceutica nombre_rol, tipo_rol, modalidad, ium
DATE	fecha_expedicion, fecha_vencimiento fecha_activo, fecha_inactivo
NUMERIC	cantidad_cum, cantidad
CATEGORICAL	estado_cum

Tabla 5. Esquema de la base de datos de Medicamentos de Venta Libre.

Tipo de dato	Columna
STRING	principio_activo concentracion, forma_farmaceutica

Tabla 6. Esquema de la base de datos de Precios de Medicamentos.

Tipo de dato	Columna
STRING	principio_activo, unidad_de_dispensacion concentracion, unidad_base, nombre_comercial fabricante, factores_precio
NUMERIC	precio_por_tableta
INTEGER	numero_factor

3.4.3 Diseño del Prompt y extracción de intención. Uno de los principales componentes de este módulo es el *Prompt* que contiene las instrucciones con las que el modelo debe trabajar. A partir de la información contenida en el trabajo de Rajkumar et al.[37] el diseño del *Prompt* se llevó a cabo de la siguiente manera:

[37]Nitarshan Rajkumar, Raymond Li y Dzmitry Bahdanau. «Evaluating the text-to-sql capabilities of large language models». En: *arXiv preprint arXiv:2204.00498* (2022).

- **Definición del rol:** Se designa al modelo como un clasificador de intención para un sistema Text-To-SQL.
- **Inyección de esquema:** Se le suministra al modelo el detalle técnico de las base de datos, junto con sus respectivos tipos de datos. (Ver tablas 4, 5 y 6).
- **Restricciones de salida:** Se instruye al modelo a atender las reglas de contenido y de salida.

El anexo B muestra un ejemplo del prompt utilizado en este trabajo, en él se define el formato en el que el modelo debe generar la intención. En este caso se define que la estructura que se requiere es un objeto en formato JSON pues es un elemento con el cual se podrá trabajar durante la fase de posprocesamiento, como se menciona en la Sección 3.5.

El resultado del proceso estocástico es un objeto estructurado que representa la intención del usuario. Esta Representación Intermedia (IR) desacopla la semántica de la sintaxis SQL final. (Ver anexo A)

3.5 DESARROLLO DEL MÓDULO DETERMINISTA

Una vez generada la intención estructurada, el proceso se encamina hacia un proceso determinista, en donde la precisión está determinada por reglas algorítmicas rígidas las cuáles se describen a continuación.

3.5.1 Normalización y validación de Intención.

En esta fase se valida el objeto JSON en contraste con el esquema real de la base de datos. Este proceso realiza las siguientes tareas:

- **Validación de atributos:** Se verifica que cada campo en *dimensions* y *filters* exista en la base de datos, también se corrigen errores de escritura mediante el diccionario semántico.
- **Tipado de datos:** Esta tarea permite asegurar que las operaciones lógicas sean compatibles con el tipo de dato.

3.5.2 Mapeo de operandos. Esta segunda fase asigna operandos al objeto JSON con base en el prompt ingresado por el usuario. Este proceso utiliza un diccionario de operandos, que mapea palabras clave que puede ingresar el usuario, para ser convertidas inmediatamente en operandos válidos. Esta fase es particulamente relevante para el manejo de preguntas con filtros temporales.

3.5.3 Ensamblaje de consultas SQL. El paso final es la traducción de intención normalizada a una sentencia *SQL* ejecutable. Al delegar esta tarea a un módulo basado en reglas, se evita el riesgo de errores sintácticos comunes en los LLMs. El algoritmo construye la sentencia SQL en el siguiente orden:

1. **SELECT:** Mapea las dimensiones y funciones de agregación.
2. **FROM:** Inyecta la tabla correspondiente.
3. **WHERE:** Indexa las condiciones de filtrado.
4. **GROUP BY** / **ORDER BY:** Aplica ordenamiento y agrupamiento

3.6 DESARROLLO E INTEGRACIÓN DEL PROTOTIPO DE SOFTWARE

Una vez consolidado el motor de traducción *Text-To-SQL*, se conformó un prototipo de *software* que permitió la interacción del usuario final con el sistema. Este prototipo usa una interfaz gráfia (GUI por sus siglas en inglés) que permite al usuario realizar preguntas semátincas en un sistema simple y de baja complejidad.

Para la construcción del *Backend* de esta GUI, se utilizó la herramienta *Flask* basada en Python desde la cuál se configuró la conexión al sistema Text-To-SQL, así como a la base de datos requerida en cada ocasión. Este marco de trabajo brinda al usuario una integración directa con los módulos del sistema *Text-To-SQL* desde la comodidad de una interfaz web.

3.7 ADAPTACIÓN DE MÉTRICAS DE DESEMPEÑO TEXT-TO-SQL

Para cuantificar el desempeño del modelo propuesto con su base de datos ante otros modelos que fueron entrenados con otras bases de datos se realizaron experimentos en los que se calcularon métricas estándar en el márco de la evaluación Text-To-SQL, como lo es el EX Score y la evaluación por componentes.

Por otra parte, en este proyecto se aplicó la métrica *WF1 Score* expuesta en la sección 2.6 utilizando unos valores de pesos $W_S = 0{,}4$, $W_W = 0{,}3$, $W_O = 0{,}2$, $W_A = 0{,}1$. Dicha métrica fue utilizada para comparar la capacidad de los diferentes modelos locales propuestos como motor para el sistema *Text-To-SQL* y seleccionar el más adecuado.

Las evaluaciones y resultados encontrados se muestran en el siguiente capítulo.

3.8 COMPLEJIDAD ESTRUCTURAL

El portal Datos Abiertos Colombia ofrece las siguientes alternativas para acceder a los datos abiertos:

- Interfaz web nativa del portal Datos Abiertos Colombia.
- Acceso con Excel a través de OData.
- Acceso programático vía API de Socrata utilizando Python.

En esta fase metodológica se establece el proceso de aplicación de las métricas de complejidad: *Métrica de Yaqin*[38], *Complejidad Ciclomática* y *Complejidad de Procesos*[39], que incluye el diseño de los diagramas de proceso y la posterior evaluación en cada métrica. En este sentido se diseñaron cuatro diagramas de proceso, uno por

[38]Yaqin, Sarno y Rochimah, «Measuring Scalable Business Process Model Complexity Based on Basic Control Structure».

[39]Benavides et al., «Lithops-HPC: Extending the Serverless Paradigm to High-Performance Computing for Accessible Resource Management».

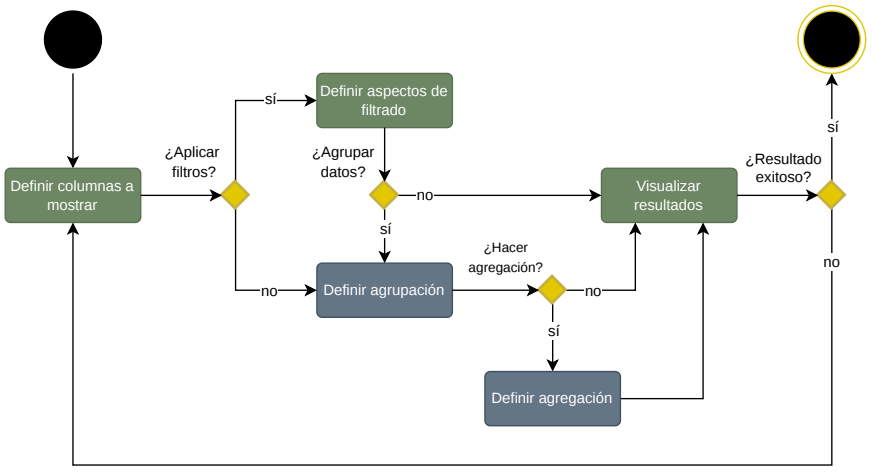
cada alternativa de acceso, también se incluyó el diagrama del proceso del prototipo propuesto en este trabajo.

3.8.1 Diagramas de procesos para métodos de acceso a bases de datos.

3.8.1.1 Acceso a través de interfaz web.

La Figura 5 detalla el proceso de navegación por la interfaz web, desde el planteamiento de las columnas sobre las que desea hacer las consultas, pasando por las secciones de toma de decisiones relacionadas con filtros y funciones de agrupamiento, hasta realizar este proceso de forma iterativa en caso de no conseguir resultados adecuados. En esta alternativa el usuario se ve limitado en el alcance de las consulta que puede hacer debido a su interfaz rígida y limitada.

Figura 5. Diagrama de actividades del método de acceso a las bases de datos mediante la interfaz web del portal Datos Abiertos Colombia.



En la Sección 2.7.3, se describió que para el cálculo de la Complejidad de Procesos es imperativo establecer un rango de valores que representan el peso cognitivo que el usuario tiene al realizar una determinada actividad dentro de los flujos de procesos. En la Tabla 7 se describe la asignación de pesos para el escenario de acceso vía web. Se asignó un peso básico ($w = 1$) a la definición de columnas por ser una interacción estándar de interfaz visual al igual que a la visualización de resultados.

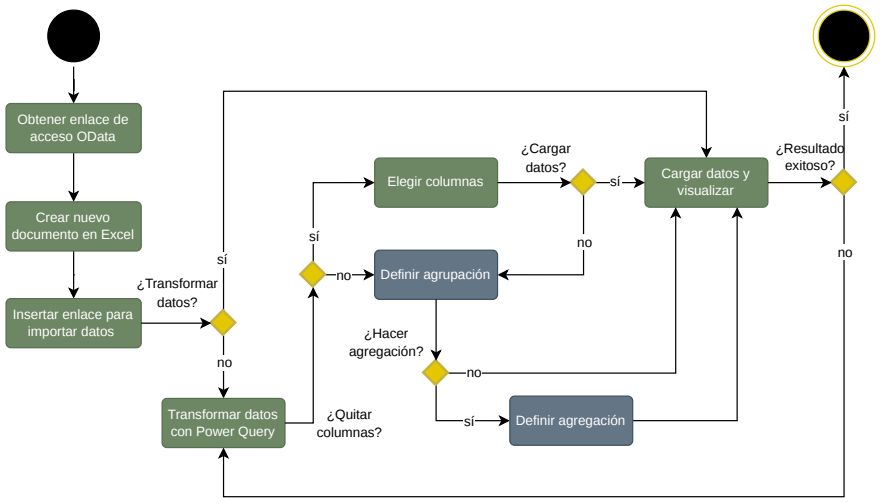
Sin embargo, las tareas de filtrado, agrupación y agregación fueron penalizadas con un peso intermedio ($w = 2$), dado que exigen al usuario comprender la lógica procedimental de la base de datos subyacente (operadores lógicos, matemáticas de conjuntos).

Tabla 7. Carga cognitiva para métrica de Complejidad de Procesos (CH) del método de acceso por interfaz Web.

Actividad / Decisión	**Peso Cognitivo**
Definir columnas a mostrar	1
Definir aspectos de filtrado	2
Definir agrupación	2
Definir agregación	2
Visualizar resultados	0

3.8.1.2 Acceso a través de Excel con OData. La Figura 6 define el proceso que tiene como fin realizar la consulta de la base de datos con Excel. Este proceso lleva al usuario desde la obtención del conjunto de datos con *OData*, un protocolo para conectar y obtener datos directamente en hojas de *Excel*. Luego pasa por el proceso de limpieza de tipos de datos, columnas y registros nulos con la herramienta *Power Query* para al final tener la hoja de cálculo resultante. Este proceso implica una serie de decisiones y también procesos iterativos si el resultado no es el esperado.

Figura 6. Diagrama de actividades del método de acceso y transformación de datos importando un archivo en Excel.



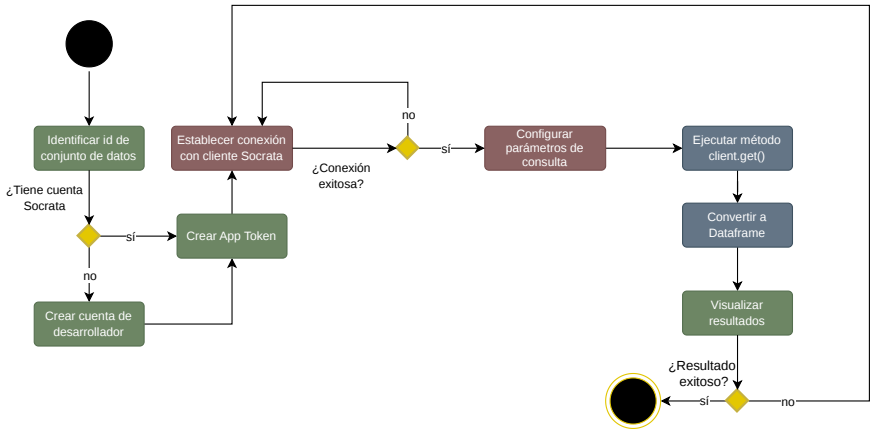
La justificación de los pesos en la Tabla 8 refleja la transición de un entorno gráfico a uno de manipulación de datos. Mientras que obtener enlaces y crear documentos son tareas básicas de interfaz ($w = 1$ y $w = 0$), la transformación de datos mediante Power Query recibe la máxima penalización ($w = 3$). Esto se debe a que exige al usuario un dominio de Excel y Power Query. Las tareas de definir agregación y agrupación siguen siendo de alta carga cognitiva pues los usuarios deben conocer como se estructuran estos parámetros para obtener buenos resultados.

Tabla 8. Carga cognitiva de métrica de Complejidad de Procesos (CH) del método de acceso mediante Excel y OData.

Actividad / Decisión	**Peso Cognitivo**
Obtener enlace OData	1
Crear documento Excel	0
Insertar enlace de datos	1
Transformar con *Power Query*	3
Elegir/Quitar columnas	1
Definir agrupación	2
Definir agregación	2
Cargar datos y visualizar	0

3.8.1.3 Acceso a través de API. La Figura 7, representa el flujo de acceso programático. Este modelo describe la secuencia de pasos técnicos requeridos para consumir los datos mediante *scripts* externos. El diagrama estructura actividades como la generación de credenciales (*App Tokens*), la configuración del entorno de desarrollo, la redacción del código (por ejemplo, en Python), la parametrización de la petición *HTTP* con sintaxis *SoQL* y la recepción del objeto JSON de respuesta.

Figura 7. Diagrama de actividades del método de acceso programático utilizando la API de Socrata.



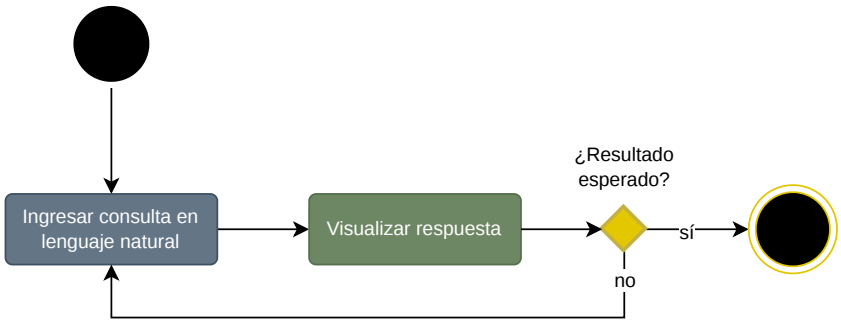
El acceso mediante API descrito en la Tabla 9 presenta la mayor concentración de pesos avanzados ($w = 3$ y $w = 2$). Se justifica la asignación del peso máximo a la conexión del cliente Socrata y la configuración de parámetros, ya que requieren explícitamente la redacción de código en lenguajes como Python, el manejo de protocolos HTTP y la estructuración de sentencias SoQL. Asimismo, la autenticación (App Token) y la manipulación de Dataframes exigen gestión de configuraciones y transformación de estructuras de datos ($w = 2$). Esto convierte a este método en un enfoque exclusivo para perfiles de ingeniería, reflejando una alta complejidad ciclomática.

Tabla 9. Carga cognitiva de métrica de Complejidad de Procesos (CH) para la conexión mediante API.

Actividad / Decisión	**Peso Cognitivo**
Identificar ID del dataset	1
Crear cuenta de desarrollador	1
Crear App Token	2
Conexión cliente Socrata	3
Configurar parámetros de consulta	3
Ejecutar método *client.get()*	2
Convertir a *Dataframe*	2
Visualizar resultados	0

3.8.1.4 Acceso a través de método propuesto. La Figura 8 ilustra la arquitectura de interacción del prototipo desarrollado en esta investigación. El modelo simplifica el dominio del usuario, estableciendo como evento de inicio la formulación de una consulta en lenguaje natural en la caja de texto de la interfaz. El diagrama describe eventos sencillos,transfiriendo el control de las operaciones lógicas y de validación sintáctica hacia los módulos estocástico y determinista culminando en la visualización de los resultados de la consulta.

Figura 8. Diagrama de actividades del método de acceso utilizando el prototipo *Text-To-SQL* desarrollado.



La evaluación del prototipo Text-To-SQL descrito en la 10 evidencia la drástica reducción de la carga de cognitiva al usuario. Se justifica la asignación de un único peso básico ($w = 1$) para el ingreso de la consulta, ya que representa la forma de

comunicación humana natural sin requerimientos técnicos. El resto de la carga estructural, relacional y procedimental es asumida en su totalidad por la arquitectura estocástica y determinista del sistema.

Tabla 10. Carga cognitiva de métrica de Complejidad de Procesos (CH) del prototipo propuesto.

Actividad / Decisión	**Peso Cognitivo**
Ingresar consulta en lenguaje natural	1
Visualizar respuesta	0

4. RESULTADOS

En este capítulo se presentan los resultados cuantitativos y cualitativos de la arquitectura propuesta. Aquí se evalúan las métricas de exactitud de ejecución global de consultas SQL (*EX score*) y se comparan con los propuestos en el estado del arte. Además, también se evalúa la precisión de la extracción semántica a nivel de cláusulas mediante la métrica *F1 score*. Finalmente se cuantifica la reducción de la complejidad del sistema propuesto mediante el prototipo de *software* haciendo uso de las métricas de complejidad estructural.

4.1 PREPROCESAMIENTO Y CARACTERIZACIÓN

La tabla 11 muestra las características técnicas de las tres bases de datos empleadas en este trabajo.

Tabla 11. Caracterización técnica de los conjuntos de datos sin pre-procesar.

Conjunto de datos	Registros	Columnas	Tipos de columnas
CUMV	159.955	29	24 de texto, 5 numéricas
PM	12.534	12	9 de texto, 3 numéricas
LMVL	997	3	3 de texto

La tabla 12 muestra los resultados del preprocesamiento, aquí se evidencian las características de los conjuntos de datos después de aplicar las diferentes etapas de limpieza. En algunas bases de datos se presenta un reducción del número de columnas porque estas contienen información redundante o completamente irrelevante (Ej. IDs).

Tabla 12. Caracterización técnica de los conjuntos de datos después de preprocesar.

Conjunto de datos	Registros	Columnas	Tipos de columnas
CUMV	158.702	23	14 de texto, 5 numéricas, 4 de fecha
PM	12.534	10	8 de texto, 2 numéricas
LMVL	997	3	3 de texto

4.2 EVALUACIÓN Y SELECCIÓN DEL MODELO

Para la evaluación de los tiempos de ejecución y consumo de memoria con *WF1 Score* se utilizó una máquina de pruebas *Acer Aspire A315-42* con una CPU *AMD Ryzen 7 3700U* de 8 núcleos a 2.30 GHz y con memoria *RAM* de 16GB DDR4, empleando un sistema operativo basado en un kernel *Linux 6.19.12*.

La tabla 13 muestra el desempeño en términos de la métrica *WF1* (explicada en la sección 2.6.2) así como el tiempo de ejecución y el consumo de memoria para los cinco modelos LLMs estudiados. Todos ellos fueron ejecutados de manera local utilizando el esquema de las bases de datos propuestas y un *prompt* estándar. (Ver anexo B).

Tabla 13. Resultados WF1 de modelos LLM propuestos

Modelos	Consumo de memoria	Tiempo de ejecución (m)	WF1 Score
Llama 3.1	4.9GB	11.39	**0.54**
Llama 3	4.9GB	11.20	0.44
Phi3	2.2GB	11.97	0.21
Qwen	4.7GB	11.37	0.52
Gemma2	1.6GB	11.71	0.17

Como se observa en la tabla 13, los modelos *Llama 3.1* y *Qwen* demostraron un desempeño significativamente superior en comparación con los modelos más ligeros como Gemma 2 y Phi-3. Esta diferencia de rendimiento se explican por el número de parámetros de cada modelo, donde se evidencia que los mejores modelos son aquellos que cuentan con un mayor número de parámetros.

La tabla 14 muestra el desempeño de los cinco modelos luego de refinar cada prompt y establecer mejoras post-procesamiento como; manejo de operandos, creación de alias semántico y normalización de datos tipo fecha. (Ver anexos C y D)

La optimización del *Pipeline* reveló una diferencia importante en la capacidad de los modelos para seguir instrucciones. Mientras que *Llama 3.1* presentó una mejoría esperada, el modelo *Qwen* elevó su rendimiento de forma considerable. La capacidad de *Qwen* para maximizar su score con menos parámetros determinó que se convirtiera en el modelo seleccionado para el sistema propuesto. En general, la ta-

Tabla 14. Resultados WF1 de modelos con esquema y *Pipeline* optimizado

Modelos	Consumo de memoria	Tiempo de ejecución (m)	WF1 Score
Llama 3.1	4.9GB	11.40	0.75
Llama 3	4.9GB	11.18	0.61
Phi3	2.2GB	11.90	0.43
Qwen	4.7GB	11.39	**0.89**
Gemma2	1.6GB	11.94	0.21

bla 14 indica que el modelo *Qwen* es el que presenta mejor relación entre el *WF1 Score*, el tiempo de ejecución y el consumo de memoria.

4.3 EVALUACIÓN GLOBAL DEL MODELO Text-To-SQL

En esta sección se compara el modelo Text-To-SQL obtenido con otros modelos Text-To-SQL del estado del arte. Si bien cada modelo Text-To-SQL está entrenado para un conjunto de datos específicos, es posible comparar los resultados de manera global de cada modelo para su base de datos específica.

4.3.1 Evaluación de exactitud global (EX Score).

La tabla 15 muestra el desempeño en términos de la métrica *EX Score* explicada en la sección 2.6.1.

Tabla 15. Resultados EX Score para 5 modelos Text-To-SQL.

Modelo	Peso	EX Score	EX/Size ratio
DAIL-SQL (GPT-4)	$\sim$1.8t	86.6 %	4.81×10^{-11}
SelECT-SQL (GPT-3.5-Turbo)	>175b	84.2 %	4.81×10^{-10}
RAT-SQLB + SPR	120m	81.3 %	6.77×10^{-7}
NatSQL + SPR	175m	83.7 %	4.78×10^{-7}
Propuesto (Qwen+Schema Linking)	7b	**87.6 %**	1.25×10^{-8}

Los resultados revelan que el modelo entrenado para esta investigación presenta un adecuado desempeño para el conjunto de datos lo cuál cumple con los objetivos de este proyecto. El porcentaje alcanzado para el modelo propuesto es el mayor de todos y puede explicarse debido a la complejidad de la base de datos con la que

se evalúa cada modelo. Además, el sistema Text-To-SQL de este trabajo introduce una arquitectura híbrida que añade un módulo con varios procesos deterministas y el módulo estocástico clásico de los modelos del estado del arte. Lo cuál ayuda a reducir las alucinaciones del sistema.

4.3.2 Evaluación de la generación de cláusulas SQL. La tabla 16 muestra los resultados del *F1 Score* para cada una de las cláusulas de la sentencia SQL.

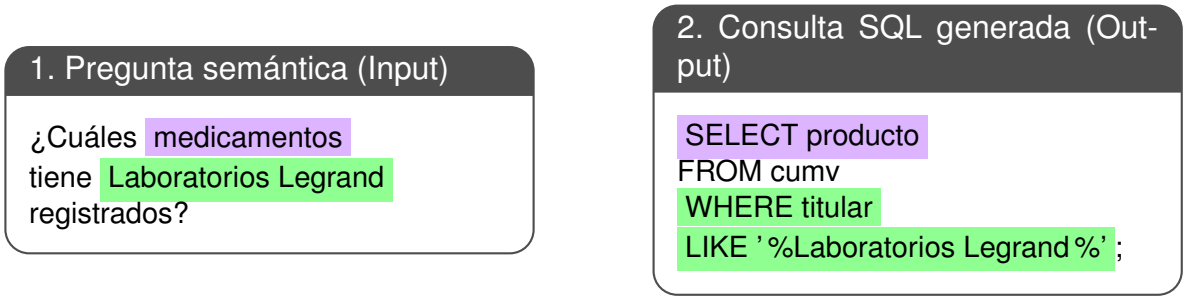
Tabla 16. Resultados F1 para modelos relevantes del estado del arte.

Modelo	**Peso**	**F1 Score**		
		SELECT	**WHERE**	**GROUP BY**
RAT-SQLB + SPR	120m	0.91	0.772	0.816
NatSQL + SPR	175m	**0.997**	**0.969**	**0.879**
Propuesto (Qwen+Schema Linking)	7b	0.966	0.924	0.871

Los resultados muestran que el modelo propuesto en este trabajo alcanza un rendimiento comparable con el estado del arte, lo cuál demuestra que incluso ante métricas más estrictas, el modelo mantiene un desempeño adecuado. Cabe resaltar los valores logrados en las cláusulas de *SELECT* y *WHERE* que son las más relevantes para realizar consultas sobre las bases de datos estudiadas en este proyecto.

4.3.3 Resultados cualitativos. En esta sección se demuestra el funcionamiento del prototipo propuesto mediante 5 preguntas semánticas representativas, que evidencian el desempeño del modelo ante los distintos tipos de consultas tales como: consultas por fecha, por campos de texto, por agrupación, entre otros.

Figura 9. Ejemplo 1. Resultados de traducción. Cláusula *SELECT* con color magenta y *WHERE* con verde.

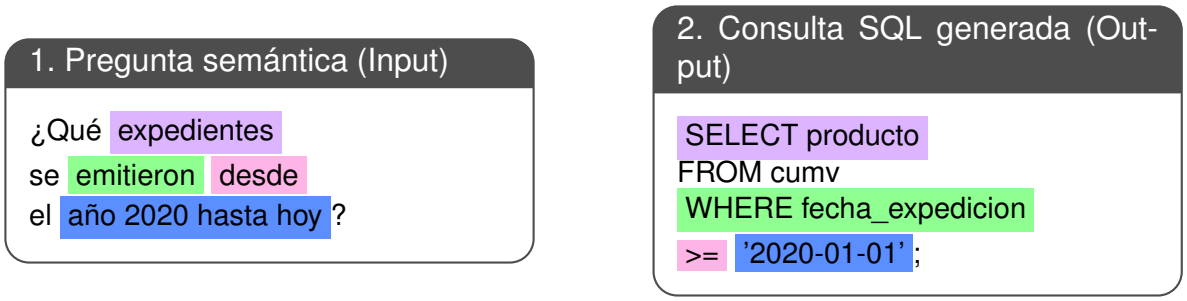


La figura 9 muestra que el modelo es capaz de reconocer las diferentes cláusulas y los valores asociados a cada una de ellas, separando la información no relevante para poder conformar la sentencia SQL adecuada. Además, se puede observar que el modelo añade adecuadamente expresiones regulares (Por ejemplo: %) ante filtros de campo de texto para adecuarse a la intención de la pregunta.

Figura 10. Ejemplo 2. Resultados de traducción. Cláusula *SELECT* con color magenta.

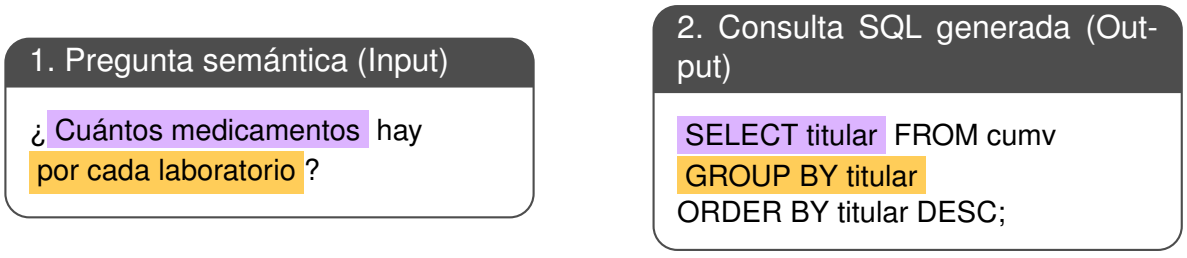
La figura 10 ilustra el caso de la cláusula *SELECT*. La que la pregunta semántica implica de manera implícita agregaciones (*COUNT*) y filtros especiales (*DISTINCT*), se puede evidenciar que el modelo realiza correctamente la traducción, siendo capaz de anidar condiciones y agregaciones dentro de la cláusula tal como la pregunta lo requería. Esta pregunta demuestra una capacidad inicial del modelo de asignar operandos que no se encuentran tan explícitos como en otras preguntas de menor complejidad.

Figura 11. Ejemplo 3. Resultados de traducción. Cláusula *SELECT* con color magenta y *WHERE* está compuesta por; la columna con verde, el operador con rosa y el valor con el azul.



La figura 11 muestra el caso de una traducción de una pregunta que contiene filtros de tipo temporal. Sin embargo, este caso es particular, ya que aparentemente la pregunta requiere un rango de fechas entre el año 2020 y el día actual para dicho filtro, pero la sentencia SQL solo contiene la condición de mayor del año 2020. Esta aparente incongruencia se puede explicar por las características de esta pregunta en específico, ya que al tratarse de la fecha de expedición no hay posibilidad de que existe alguna posterior a la fecha actual en la que se realice la consulta. En otras palabras, el filtro generado por el modelo es realmente el único indispensable para satisfacer esta consulta y añadir otro sería completamente innecesario. Este escenario demuestra que el modelo no solo es capaz de asignar correctamente filtros temporales en una cláusula *SELECT*, si no que además puede inferir y simplificar dicho filtro.

Figura 12. Ejemplo 4. Resultados de traducción. Cláusula *SELECT* con magenta y *GROUP BY* con naranja.

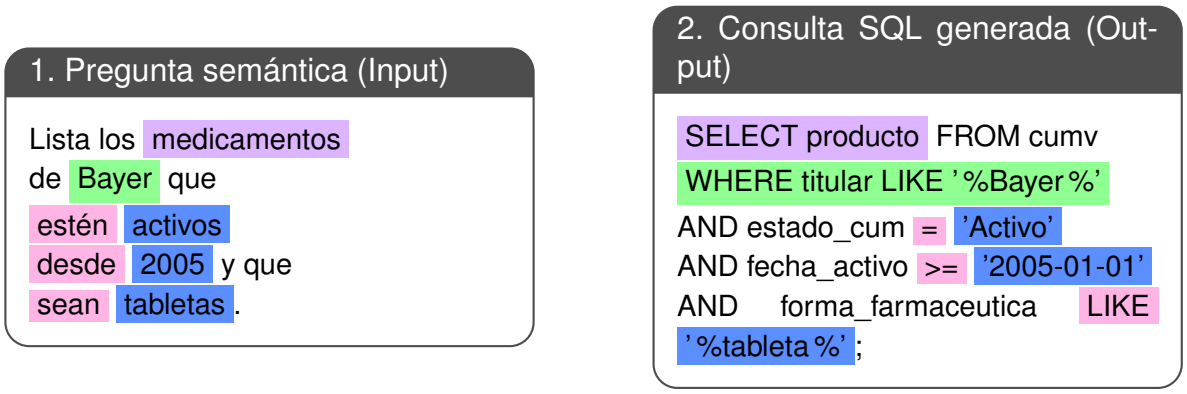


La figura 12 detalla como el modelo genera una cláusula *SELECT* incompleta así como un erroneo agrúpamiento. Este resultado inadecuado se puede explicar por la

dificultad de los modelos Text-To-SQL para generar consultas asociadas a la cláusula *GROUP BY* ya que exigen una alta capacidad de extraer y asociar información implícita de la pregunta semántica.

La figura 13 muestra el caso de preguntas extensas, de varios filtros y un elevado nivel de complejidad). En particular, la dificultad de esta pregunta radica en la información implícita escondida en la semántica del idioma. En este escenario el modelo debe identificar correctamente dos columnas distintas para realizar el filtro (*estado_cum* y *fecha_activo*) a partir de una misma palabra en la pregunta semántica (*activos*). Además, el modelo agrega correctamente varios filtros de diferentes tipos en una misma cláusula *WHERE* y añade las expresiones regulares en los filtros de texto correspondientes.

Figura 13. Ejemplo 5. Resultados de traducción. Cláusula *SELECT* con magenta y *WHERE* está compuesta por; la columna con verde, el operador con rosa y el valor con el azul.



4.4 EVALUACIÓN DEL MODELO FRENTE A DIFERENTES BASES DE DATOS

4.4.1 Análisis cuantitativo. El desempeño del modelo fue evaluado usando otras bases de datos del sector salud. Específicamente, se realizó una comparación de rendimiento con las bases de datos Código Único de Medicamentos Vigentes (CUMV), Listado de Medicamentos en Venta Libre (LMVL) y Precios de Medicamentos (PM). La tabla 17 presenta los resultados para las métricas de *WF1 Score*, *EX Score* y *F1 Score*.

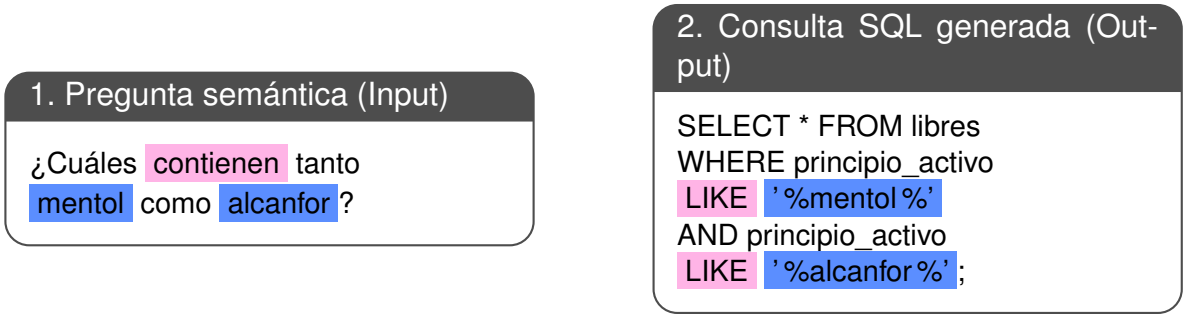
Tabla 17. Comparación de resultados F1, EX y WF1 para el modelo evaluado con diferentes bases de datos

Modelo	F1 Score			EX Score	WF1 Score
	SELECT	WHERE	GROUP BY		
Qwen + SL + CUMV	**0.966**	0.924	0.871	87.6 %	0.897
Qwen + SL + PM	0.940	**0.929**	0.853	**87.9 %**	**0.905**
Qwen + SL + LMVL	0.931	0.922	**0.893**	85.1 %	0.895

Los resultados indican que el modelo propuesto alcanza un buen desempeño general para base de datos similares. En particular, es notable que los mejores resultados para las diferentes métricas se alcanzan en los casos de *PM* y de *LMVL*, lo cuál puede explicarse debido a la menor complejidad de estas bases de datos. Estos resultados evidencian que la estrategia de entrenamiento y la configuración empleada permiten una traducción Text-To-SQL aceptable.

4.4.2 Análisis cualitativo. Esta sección presentan 3 ejemplos cualitativos; 2 correspondientes a resultados del modelo con la base de datos LMVL y 1 correspondiente a la base de datos PM.

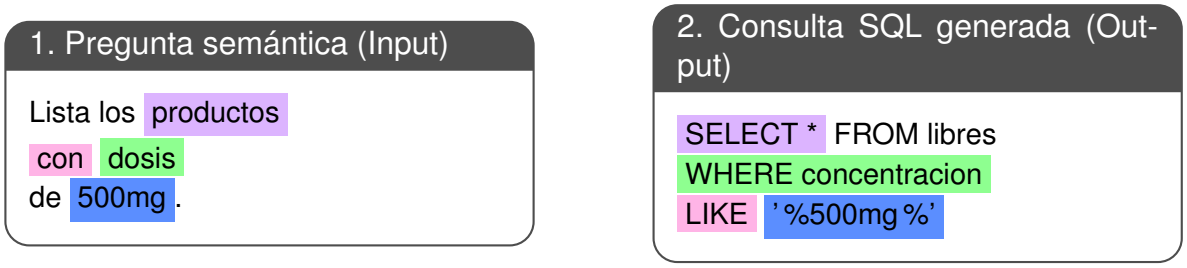
Figura 14. Ejemplo 6. Resultados de traducción para base de datos LMVL. Cláusula *WHERE* compuesta por; el operador con rosa y el valor con el azul.



La figura 14 muestra la capacidad del modelo de extraer información implícita sobre la columa *SELECT* y *WHERE*. Esto demuestra que el modelo es capaz de desempeñarse correctamente en escenarios con un contexto muy limitado y una alta ambigüedad en la pregunta semántica.

La figura 15 evidencia una capacidad asociativa del modelo para convertir expresiones semánticas como *dosis* en una columna de la cláusula *WHERE* de nombre *concentración*. Este ejemplo se explica por el diseño detallado del esquema para la base de datos que permite al modelo traducir correctamente expresiones distintas dado el contexto adecuado. En contraste, el modelo seleccionó innecesariamente todas las columnas disponibles de la tabla. Si bien este caso no es un error, ya que el modelo obtuvo efectivamente la información requerida, la sentencia generada accede a información no relevante para la pregunta.

Figura 15. Ejemplo 7. Resultados de traducción para base de datos LMVL. Cláusula *SELECT* con magenta, *WHERE* está compuesta por; la columna con verde, el operador con rosa y el valor con el azul.



La figura 16 ilustra el ejemplo cualitativo relacionado con el desempeño del modelo con la base de datos PM. El modelo muestra una capacidad para traducir expresiones con diversos tipos de filtros e información implícita sobre algunos de ellos. En este caso el modelo es capaz de reconocer el nombre de la columna que se aplica al primer filtro (*fabricante*) a pesar de dicha información se omite en la pregunta semántica. Además, el modelo aplica correctamente filtros numéricos de precios en la cláusula *WHERE*. Sin embargo, al igual que en el ejemplo de la base de datos anterior (Fig. 15), el modelo asigna columnas innecesarias en la cláusula *SELECT*.

Figura 16. Ejemplo 8. Resultados de traducción para base de datos PM. Cláusula *SELECT* con magenta, *WHERE* está compuesta por; la columna con verde, el operador con rosa y el valor con azul.

1. Pregunta semántica (Input)

Muestra los medicamentos de Genfar con precio mayor a 300.

2. Consulta SQL generada (Output)

```
SELECT * FROM precios
WHERE fabricante LIKE
'%Genfar%' AND
precio_por_tableta >300
```

4.5 EVALUACIÓN DE COMPLEJIDAD ESTRUCTURAL

En esta sección se presentan los resultados de la evolución de complejidad estructural para las cuatro alternativas de consulta propuestas. Con el fin de facilitar la lectura de este libro, los diagramas descritos en la sección 3.8, se presentan nuevamente en este apartado con un número sobre cada bloque de actividad que indica el peso cognitivo asignado con base en la complejidad de la actividad

Figura 17. Diagrama de actividades para el acceso vía interfaz web nativa.

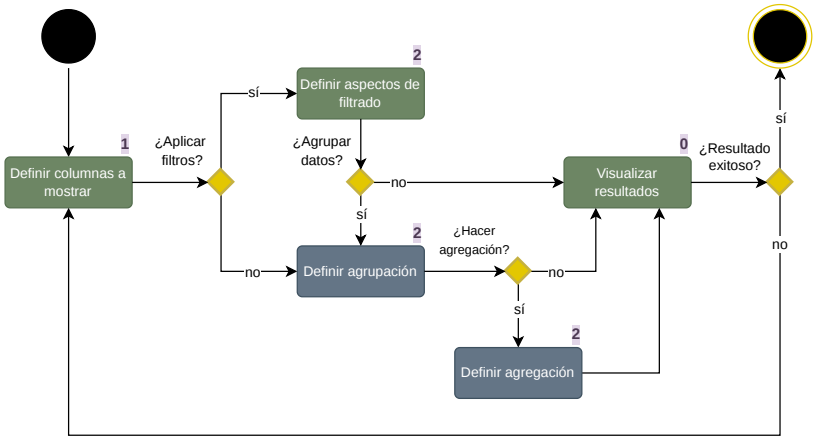


Figura 18. Flujo de actividades para la importación y transformación en Excel.

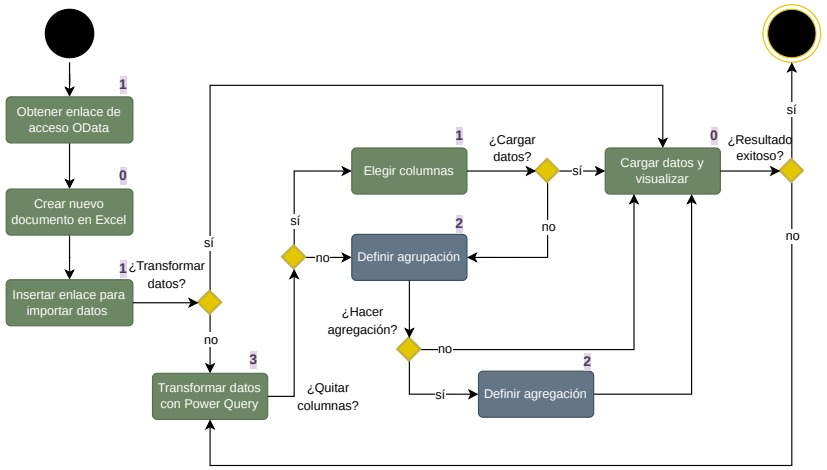


Figura 19. Diagrama de actividades para el acceso programático vía API.

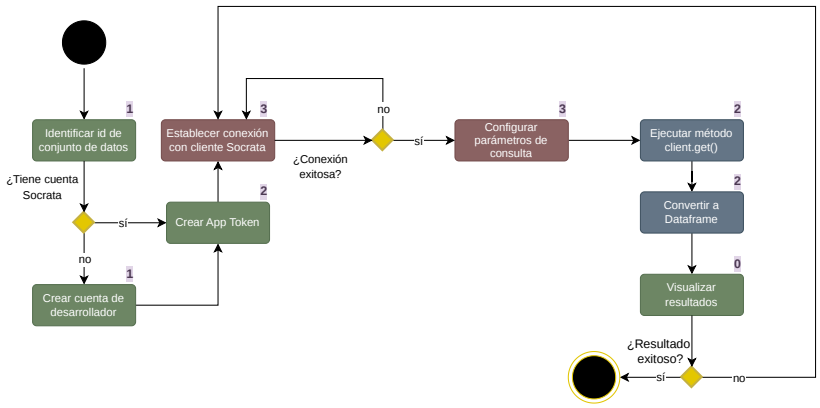
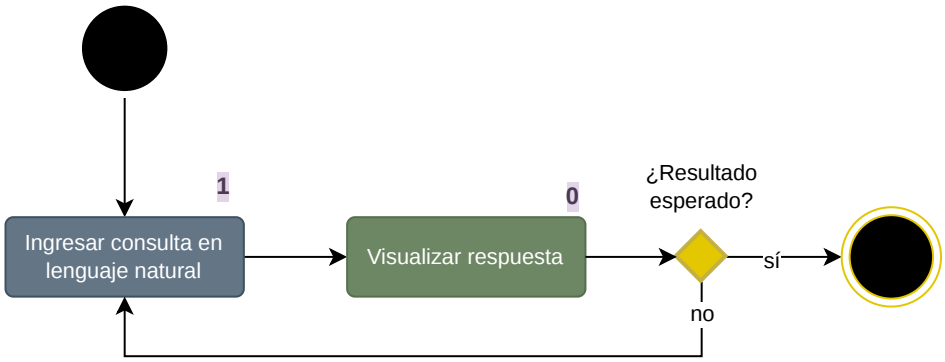


Figura 20. Flujo de interacción del prototipo *Text-To-SQL*.



La Tabla 18 resume el conteo de elementos de cada diagrama. Esta evidencia que el método *Excel* es el más denso en términos de Nodos (15) y Arcos (20), penalizado por las bifurcaciones requeridas en el preprocesamiento de datos.

Tabla 18. Conteo de elementos presentes en los diagramas de procesos.

Parámetro de Yaqin	Web	Excel	API	Prototipo (NLP)
Nodos (Ns)	11	15	13	5
Arcos (As)	14	19	15	5
Bifurcaciones (XOR)	4	5	3	1
Bucles	1	2	2	1

En la Tabla 19 se muestran los resultados relacionados con el cálculo de la profundidad del proceso de la métrica de *Yaqin*. Se evidencia que el acceso vía API refleja el promedio de profundidad (D_{avg}) más alto (5.9), lo que demuestra que el acceso programático exige atravesar múltiples capas de validación antes de obtener un resultado.

Tabla 19. Resultados de promedio de profundidad (D_{avg}) en diagramas de procesos.

Parámetro de Yaqin	Web	Excel	API	Prototipo (NLP)
Actividades (Acs)	5	8	8	2
Profundidad total	18	32	47	3
Promedio de profundidad (D_{avg})	3.6	4.0	5.9	1.5

Nota: El promedio de profundidad se calcula mediante $D_{avg} = \frac{\sum_{j=1}^{Acs} D_j}{Acs}$, donde el resultado obtenido permite determinar la complejidad de profundidad final según la expresión $C_D = CW_{C_D} \cdot D_{avg}$.

En este caso la alternativa *API de Socrata* refleja el promedio de profundidad más alto (5.9). Esto demuestra que el acceso programático exige atravesar múltiples capas de validación lógica antes de obtener el primer resultado visible. En la Tabla 20 se resume el cálculo ponderado de cada parámetro y el resultado final de la métrica de Yaqin.

Tabla 20. Consolidado de Complejidad de Interacción (YC) para cada alternativa de acceso.

Parámetro	Web	Excel	API	Prototipo (NLP)
Nodos (Ns)	11	15	13	5
Arcos (As)	14	19	15	5
Complejidad cíclica (C_{cyc})	3	6	6	3
Complejidad de bifurcación (C_{XOR})	24	30	18	6
Complejidad de profundidad (C_Ds)	50.4	56	82.6	21
Complejidad de Yaqin (YC)	102.4	126	134.6	40

Nota: El valor consolidado se determinó utilizando la ecuación de Yaqin: $YC = Ns + As + C_{AND} + C_{XOR} + C_{OR} + C_{bucle} + C_D$.

La Tabla 21 muestra la comparación integral de todas las métricas. Los resultados validan que el prototipo *Text-to-SQL* logra reducir la complejidad de interacción (YC) a un valor de 40, lo que representa una disminución del $60\,\%$ respecto a la interfaz web nativa. Esta reducción es consistente con la caída drástica en la complejidad del proceso (CH), que pasa 14 para la alternativa API y 1 para el enfoque propuesto.

Tabla 21. Consolidado de métricas de complejidad estructural y de proceso.

Métrica	**Web**	**Excel**	**API**	**Prototipo (NLP)**
Complejidad Ciclomática (CC)	5	6	4	2
Complejidad del proceso (CH)	7	10	14	1
Complejidad de Yaqin (YC)	102.4	127.0	134.6	40.0

5. CONCLUSIONES

Este trabajo presenta un prototipo de software capaz de procesar preguntas en lenguaje natural para generar consultas SQL. El prototipo tiene una arquitectura que integra un módulo estocástico y uno módulo determinista. Esta característica es clave en el proceso ya que permite delegar la extracción de la intención semántica a un LLM como *Qwen 3* y el ensamblaje de la consulta SQL a un módulo con reglas programáticas, esta división logra desacoplar con éxito la interpretación del lenguaje natural, de la sintaxis SQL y permite mitigar las alucinaciones y generar consultas SQL válidas.

Los resultados cuantitativos demostraron el buen desempeño del prototipo para las bases de datos estudiadas. Además, permitieron observar que el desempeño del módulo estocástico está directamente relacionado con la calidad del esquema creado. Esto se hizo evidente durante la evaluación comparativa, donde el modelo alcanzó su mejor desempeño en las métricas *F1-Score (WF1)* y *Execution Score (EX)* con el la base de datos Precios de Medicamentos (PM). Esto puede explicarse debido a su esquema extenso y detallado. Asimismo, el análisis de la métrica *F1 Score* para medir el desempeño del modeLo en las cláusulas (*SELECT*, *FROM*, *WHERE*) demuestra que los mejores resultados fueron más repartidos entre las bases de datos ya que esta métrica evalúa con mayor detalle el desempeño para cada una de las cláusulas.

La validación del sistema sobre estas diferentes bases de datos demostró su capacidad de adaptación gracias a un *prompt* correctamente configurado y a un post-procesamiento robusto, mostrando la flexibilidad del modelo para adaptarse a nuevos conjuntos de datos, garantizando un acceso fluido a la información.

La integración de un módulo determinista para realizar el post-procesamiento de la intención, sumado a la interfaz de usuario, permitió reducir significativamente la complejidad de acceso y consulta a conjuntos de datos abiertos del sector salud y protección social. Además, debido a que el prototipo elimina la necesidad de dominar el lenguaje SQL y otras tecnologías por parte del usuario final al ofrecer una experiencia de interacción simplificada frente a los métodos convencionales, se pudo corrobrar su beneficio frente a escenarios tradicionales, esto pudo evidenciarse

mediante métricas de complejidad. En conclusión, en este trabajo se logró responder la pregunta de investigación y cumplir con los objetivos planteados mediante el desarrollo de un sistema de interfaz natural Text-To-SQL para la consulta de bases de datos públicas del sector salud, con un desempeño adecuado y comparable con el de los actuales modelos del estado del arte, capaz de ofrecer un acceso simple a los datos, disminuyendo el nivel de complejidad frente a los métodos de consulta existentes.

6. TRABAJO FUTURO

Como se observó, el modelo de esta investigación aún tiene un gran margen porcentual de mejora en las tres méticas Text-To-SQL evaluadas (EX Score, F1 Score y WF1 Score). Este compoportamiento puede atribuirse principalmente a un pequeño grupo de preguntas de alta complejidad que involucran distintos tipos de filtros, agregaciones y agrupaciones en una misma sentencia las cuales aún no pueden ser totalmente resueltas por los sistemas aquí expuestos. Debido a esto, como trabajo futuro sería interesante crear nuevos mecanismos en el módulo determinista que permitan maximizar el rendimiento ante este tipo de preguntas complejas sin comprometer el desempeño ante las más simples.

A su vez, el modelo actualmente depende de la creación de un detallado esquema de la base de datos para funcionar correctamente. A futuro sería interesante crear un módulo de reconocimiento y generación automática de esquemas que facilite aún más la adaptación y despliegue del modelo con nuevas bases de datos. En este sentido, sería valioso investigar la extensión del modelo a otras bases de datos fuera del sector de salud y protección social e incluso evaluar su generalización ante diferentes idiomas, ampliando así su aplicabilidad y robustez.

BIBLIOGRAFÍA

Arana, Carlos. *Redes neuronales recurrentes: Análisis de los modelos especializados en datos secuenciales*. Inf. téc. Serie Documentos de Trabajo, 2021.

Benavides, Andrés et al. «Lithops-HPC: Extending the Serverless Paradigm to High-Performance Computing for Accessible Resource Management». En: *Available at SSRN 6174512* ().

Cuvides López, Hillary y Abimael Guerrero Vega. *Evaluación del impacto económico del perfil epidemiológico en el listado básico de medicamentos en institución hospitalaria Universidad del Norte*. Trabajo de grado. Universidad del Atlántico, 2023. URL: https://repositorio.uniatlantico.edu.co/server/api/core/bitstreams/5c9124c5-f3dc-4bef-874e-058afff2737b/content.

Datos Abiertos Colombia. *La Plataforma De Datos Abiertos Del Gobierno Colombiano*. 2025. URL: https://n9.cl/8lun4 (visitado 27-08-2025).

— *Reportes Portal Nacional Datos Abiertos*. Datos Abiertos Colombia. 2025. URL: https://www.datos.gov.co/stories/s/Reportes-Portal-Nacional-Datos-Abiertos/pvyw-9yqs (visitado 10-09-2025).

García Serrano, Alberto. *Inteligencia Artificial. Fundamentos, práctica y aplicaciones*. Rc Libros, 2012.

International Organization for Standardization. *ISO 8601 — Date and time format*. ISO. 21 de feb. de 2017. URL: https://www.iso.org/iso-8601-date-and-time-format.html (visitado 30-03-2026).

Jeong, Geunyeong et al. «Improving Text-to-SQL with a Hybrid Decoding Method.» En: *Entropy (Basel, Switzerland)* 25.3 (2023). DOI: 10.3390/e25030513.

Katsogiannis-Meimarakis, George y Georgia Koutrika. «A survey on deep learning approaches for text-to-SQL». En: *The VLDB Journal* 32 (ene. de 2023). DOI: 10.1007/s00778-022-00776-8.

Lichtenauer, Norbert et al. «A Scoping Review on Analysis of the Barriers and Support Factors of Open Data». En: *Information* 15.1 (2024). DOI: 10.3390/info15010005. URL: https://www.mdpi.com/2078-2489/15/1/5.

Liu, Xinyu et al. «A Survey of Text-to-SQL in the Era of LLMs: Where are we, and where are we going?» En: *IEEE Transactions on Knowledge and Data Engineering* (2025).

Ministerio de Tecnologías de la Información y las Comunicaciones. *Colombia alcanza el quinto lugar mundial en apertura de datos según el Global Data Barometer 2025*. 2025. URL: https://acortar.link/g3jRpv (visitado 27-08-2025).

— *Datos Abiertos Colombia*. 2026. URL: https://www.datos.gov.co/.

Ministerio de Tecnologías de la Información y las Comunicaciones. Dirección de Gobierno Digital. *Estudio Previo General. Formato: GCC-TIC-FM-001 V16*. Inf. téc. 2025. URL: https://www.mintic.gov.co/.

Nascimento, Eduardo et al. «Text-to-SQL Meets the Real-World». En: *Proceedings of the 19th International Conference on Software Technologies (ICSOFT 2024)*. 2024, págs. 61-72. DOI: 10.5220/0012555200003690.

Pourreza, Mohammadreza y Davood Rafiei. «Din-sql: Decomposed in-context learning of text-to-sql with self-correction». En: *Advances in neural information processing systems* 36 (2023), págs. 36339-36348.

Qin, Bowen et al. «A survey on text-to-sql parsing: Concepts, methods, and future directions». En: *arXiv preprint arXiv:2208.13629* (2022).

Rajkumar, Nitarshan, Raymond Li y Dzmitry Bahdanau. «Evaluating the text-to-sql capabilities of large language models». En: *arXiv preprint arXiv:2204.00498* (2022).

Suave, André Augusto. *Inteligência Artificial*. ProQuest Ebook Central. Freitas Bastos, 2024. URL: https://ebookcentral.proquest.com/lib/bibliouis-ebooks/detail.action?docID=31498447.

Vaswani, Ashish et al. «Attention is all you need». En: *Advances in neural information processing systems* 30 (2017).

Xu, Wenbo et al. «TS-SQL: Test-driven Self-refinement for Text-to-SQL». En: *Findings of the Association for Computational Linguistics: EMNLP 2025*. 2025.

Yaqin, Muhammad Ainul, Riyanarto Sarno y Siti Rochimah. «Measuring Scalable Business Process Model Complexity Based on Basic Control Structure». En: *International Journal of Intelligent Engineering and Systems* 13 (dic. de 2020), págs. 52-65. DOI: 10.22266/ijies2020.1231.06.

Yu, Tao et al. «Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task». En: *ArXiv* abs/1809.08887 (2018). URL: https://api.semanticscholar.org/CorpusID:52815560.

Zhong, Victor, Caiming Xiong y Richard Socher. *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning*. 2017. arXiv: 1709.00103 [cs.CL]. URL: https://arxiv.org/abs/1709.00103.

ANEXOS

Anexo A. SALIDA DEL MODELO LLM

El listado A.1 presenta un ejemplo de la salida generada por el modelo. Proceso descrito como extracción de la intención.

Listado A.1. Ejemplo de output del modelo LLM (intención).

```
{
  "intent_type": "search",
  "dimensions": ["producto"],
  "filters": [
    {
      "column": "principio_activo",
      "operator": "=",
      "value": "paracetamol"
    }
  ],
  "ordering": [],
  "limit": 10
}
```

Anexo B. PROMPT BÁSICO

El listado B.1 muestra un primer prompt con el que se evaluó inicialmente a los modelos LLMs.

Listado B.1. *Prompt* inicial generado para evaluación.

```
prompt = (
    "Eres un clasificador de intencion para un sistema Text-to-SQL. "
    "Devuelve SOLO JSON valido. No expliques la respuesta. "
    f"Pregunta del usuario: {pregunta} "
    "Devuelve JSON con: intent_type, metrics, dimensions,\
    filters, ordering, limit."
)
```

El listado C.1 evidencia el prompt avanzado que se obtuvo después de múltiples iteraciones y mejoras para evaluar los modelos LLMs.

Listado C.1. *Prompt* avanzado generado para evaluación.

```
system_prompt = f"""
Eres un clasificador de intencion para un sistema Text-to-SQL.

Debes analizar la pregunta del usuario y devolver una representacion
   estructurada
de la intencion de consulta.

REGLAS IMPORTANTES:
- Solo puedes usar columnas que existan en el esquema.
- No inventes columnas.
- Devuelve SOLO JSON valido.
- No expliques nada.
- No devuelvas texto fuera del JSON.
- NUNCA inventes filtros.

ESQUEMA DE BASE DE DATOS:
{schema_description}

FORMATO DE RESPUESTA:

{{
  "intent_type": "search | aggregation",
  "metrics": [],
  "dimensions": [],
  "filters": [
    {{
      "column": "column_name",
      "operator": "=",
      "value": "value"
    }}
  ],
  "ordering": [
    {{
      "column": "column_name",
```

```
      "direction": "asc"
    }}
  ],
  "limit": 10
}}
"""
```

Anexo D. ALIAS SEMÁNTICO

En los listados D.1, D.2 y D.3 se observa el alias creado para el mapeo de palabras clave post-generación de intención por parte del LLM en el flujo del sistema Text-To-SQL.

Listado D.1. Alias semántico creado para CUMV como mejora al sistema Text-To-SQL.

```
SEMANTIC_ALIAS = {
    "medication name": "producto",
    "expiration date": "fecha_vencimiento",
    "medicine": "producto",
    "drug name": "producto",
}
```

Listado D.2. Alias semántico creado para PM como mejora al sistema Text-To-SQL.

```
SEMANTIC_ALIAS = {
    "medicamento": "nombre_comercial",
    "medicina": "nombre_comercial",
    "producto": "nombre_comercial",
    "droga": "principio_activo",
    "precio": "precio_por_tableta",
    "costo": "precio_por_tableta",
    "laboratorio": "fabricante",
    "empresa": "fabricante"
}
```

Listado D.3. Alias semántico creado para LMVL como mejora al sistema Text-To-SQL.

```
SEMANTIC_ALIAS = {
```

```
    "medicamento": "principio_activo",
    "medicina": "principio_activo",
    "producto": "principio_activo",
    "droga": "principio_activo",
    "formula": "principio_activo",
    "composicion": "principio_activo",

    "dosis": "concentracion",
    "contenido": "concentracion",
    "potencia": "concentracion",

    "presentacion": "forma_farmaceutica",
    "forma": "forma_farmaceutica",
    "via": "forma_farmaceutica"
}
```

Anexo E. FÓRMULAS DE COMPLEJIDAD DEL PROCESO

En esta sección se detallan las ecuaciones matemáticas utilizadas para el cálculo de las diferentes métricas de complejidad del proceso evaluadas en la metodología.

Para el caso de las bifurcaciones, la complejidad de las compuertas paralelas (C_{AND}) se define como:

$$C_{AND} = CW_{And} \cdot \Sigma_i(n!)_i \tag{6.1}$$

La complejidad de bifurcaciones exclusivas (C_{XOR}) está dada por:

$$C_{XOR} = CW_{XOR} \cdot \Sigma_i(n)_i \tag{6.2}$$

La complejidad de bifurcaciones inclusivas (C_{OR}) se evalúa mediante:

$$C_{OR} = CW_{OR} \cdot \sum_i \left(\sum_{k=1}^{n-1} \cdot \frac{n!}{(n-k)!} \right)_i \tag{6.3}$$

Para las iteraciones o repeticiones dentro del flujo, la complejidad de ciclos (C_{bucle})

se calcula como:

$$C_{bucle} = CW_{bucle} \cdot \frac{Acs_{bucle}}{Dm} \tag{6.4}$$

Finalmente, para medir el desgaste cognitivo acumulado, se calcula primero el promedio de profundidad (D_{avg}):

$$Davg = \frac{\sum_{j}^{Acs} \cdot D_j}{Acs} \tag{6.5}$$

Con este promedio, se determina la complejidad de profundidad (C_D):

$$C_D = CW_{C_D} \cdot Davg \tag{6.6}$$