Batalla de Gigantes:
Comparando LLMs para la traducción de lenguaje natural a SQL

Enzo Nicolás Manolucos
Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba
enzo.manolucos@mi.unc.edu.ar

Abstract

Los grandes modelos de lenguaje (LLMs) son cada vez más eficientes en tareas que requieren conocimiento previo, como la interacción con bases de datos para almacenar, organizar y compartir información. El enfoque Text-to-SQL permite convertir instrucciones en lenguaje natural, proporcionadas por un usuario, en *querys* SQL funcionales, por lo que evaluar su rendimiento es clave. Este proyecto compara el rendimiento de Gemini 1.5 Flash y GPT-4o mini, utilizando una base de datos diseñada para la evaluación y cuatro métricas para medir la efectividad. Los resultados muestran un desempeño similar, con una ligera ventaja para Gemini 1.5 Flash, aunque GPT-4o mini sobresale en *querys* complejas con poco contexto. El código, resultados y anotaciones están disponibles en el siguiente link.

1 Introducción

Text-to-SQL es un tipo de tarea en el procesamiento de lenguaje natural (NLP) cuyo objetivo es generar automáticamente *querys* SQL a partir de texto en lenguaje natural. Esta tarea implica convertir instrucción de texto de entrada a una representación estructurada y luego utilizar esta representación para generar una *query* SQL semánticamasnte correcta que pueda ejecutarse en una base datos.

A través del uso de grandes modelos de lenguaje (LLMs), un usuario sin conocimientos o con los conocimientos mínimos de los datos puede acceder a bases de datos e interactuar con ellas mediante Text-to-SQL. Los LLMs han emergido como grandes herramientas, demostrando un gran potencial para comprender preguntas complejas y poder traducirlas de forma precisa.

Además, el uso óptimo de los *prompts* juega un papel crucial al momento de darle indicaciones a los LLMs para generación *querys* SQL. Para madurar estos modelos, se desarrollaron a lo largo de los años diversos *datasets* y métricas exclusivamente diseñadas para este tipo de tareas.

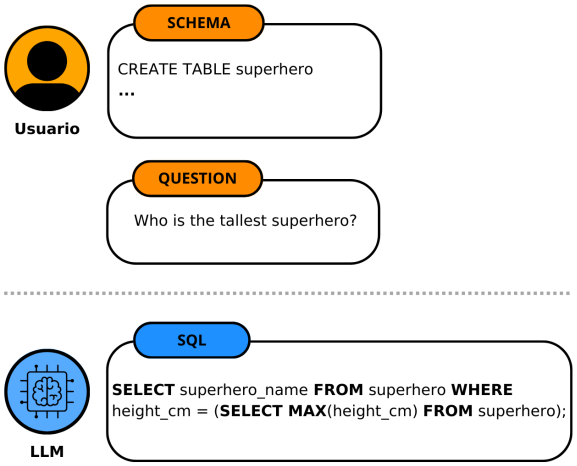

Figura 1: Ejemplo de una tarea Text-to-SQL. Un usuario brinda el esquema el cual describe la base de datos y una pregunta relacionada al mismo. El modelo toma el esquema y la pregunta como la entrada, y genera una *query* SQL como salida.

2 Trabajos Previos

2.1 Datasets

Existen diversos *datasets* para evaluar esta tarea. WikiSQL (Zhong et al., 2017) es un *dataset* que contempla operaciones simples del tipo SELECT y WHERE sin incluir *querys* anidades ni operaciones del tipo JOIN. Más tarde se desarrollo SPIDER (Yu et al., 2018) para ser una aproximación más certera a escenarios del mundo real, donde los modelos deben crear *querys* complejas y entender el esquema de la base de datos.

Recientemente[1] se creó BIRD (Li et al., 2024) para cerrar la brecha entre investigación académica y aplicaciones de la industria. Este *dataset* introduce preguntas que requieren conocimiento externo y optimización en la ejecución de las *querys*.

[1] SPIDER 2.0 no había sido lanzado al momento de iniciar este proyecto.

2.2 Métodos

Existe una diferencia significativa en la precisión entre LLMs (74.12 %) y una persona experta en el dominio (92.96 %)[2]. Debido a que aun existe una diferencia grande entre ambos, se da la necesidad de continuar con el desarrollo de los LLMs destinados a esta tarea para poder equiparar los resultados.

Se exploraron diversos enfoques para crear modelos capaces de realizar tareas Text-to-SQL. Una forma de hacerlo es mediante el *finetune* sobre LLMs en ejemplos del tipo Text-to-SQL. Este proceso implica utilizar un modelo preentrenado, la preparación de los datos, el ajuste de parámetros y la evaluación. El objetivo es generar resultados precisos y que se adapten a este tipo de tareas.

Los autores (Pourreza et al., 2024) proponen generar respuestas utilizando estrategias como el enfoque divide y vencerás, razonamiento basado en planes de ejecución y ejemplos sintéticos personalizados. Luego un agente selecciona la mejor *query* basado en comparaciones.

Los autores (Liu and Tan, 2023) proponen un nuevo paradigma llamado *Divide-and-Prompt*, basado en la técnica *Chain of Thought* (CoT), que guía al modelo mediante un proceso de razonamiento paso a paso. Este enfoque combina la división de tareas complejas en subtareas manejables con la aplicación del razonamiento CoT para resolver cada una de ellas.

3 Experimento

3.1 Dataset

El *dataset* de BIRD es usado en este proyecto ya que esta orientado a escenarios más reales y del mundo real de las bases de datos, contempla diferentes campos de dominio y presenta tres niveles de complejidad: *Simple*, *Moderate* y *Challenging*.

Contiene 12.721 preguntas con sus respectivas *querys* SQL y 95 bases de datos, con un tamaño total de 33.4 GB. Debido a su tamaño y alcance significativos, el experimento se llevó a cabo utilizando únicamente una base de datos.

Superhero incluye *querys* relacionadas al mundo de los comics con un total de 129 preguntas con su *query*, divididas en 63% *Simple*, 26% *Moderate* y 12% *Challenging*. Si bien trabajos previos (Wretblad et al., 2024) demostraron la existencia de ruido en todo de BIRD, esta base de datos presenta el menor nivel de ruido, con un 15%. En el apéndice A se describe con más detalle la base de datos.

[2]BIRD-Bench Leaderboard ingresado el 11-11-2024.

3.2 Métricas

Para evaluar el desempeño de los modelos, se decidió utilizar las siguientes métricas:

- **Execution Accuracy (EX)** mide la proporción de *querys* en los que los resultados ejecutados entre las *querys* predichas y las verdades son idénticos, en relación total de *querys* SQL.
- **Valid Efficiency Score (VES)** mide la eficiencia de las *querys* generadas válidas por los modelos. Esto se refiere a las *querys* generadas cuyos resultados coinciden con las *querys* verdaderas. La eficiencia, en este caso, se refiere al tiempo de ejecución.
- **Component Matching (CM)** mide la coincidencia exacta promedio entre los *keywords* de las *querys* generadas y las *querys* verdaderas (Yu et al., 2018).
- **Valid SQL (VA)** mide la proporción de *querys* generadas que pueden ejecutarse sin errores, indistintamente del resultado (Zhong et al., 2020).

Las dos primeras métricas son las utilizadas en el *dataset* seleccionado (Li et al., 2024). CM servirá para analizar las *querys* generadas por palabras y VA para demostrar que los modelos pueden generar *querys* sin errores.

3.3 Modelos

Los modelos utilizados son Gemini 1.5 Flash y GPT-4o mini, a través de sus APIs. La decisión de utilizar estos modelos radica en su facil acceso y su popularidad entre los usuarios, donde nace el titulo de este proyecto al comparar estos dos grandes modelos para este tipo de tareas. La Tabla 1 muestra la comparación[3] de los modelos utilizados.

Característica	Gemini 1.5 Flash	GPT-4o Mini
Entrada	1.048.576 tokens	128.000 tokens
MMLU	78.9 %	82 %
Velocidad	166 tokens/seg	97 tokens/seg
Latencia	1.06 seg	0.56 seg
Idiomas	Más de 100	No especificado
Lanzamiento	09/2024	07/2024
Entrenamiento	11/2023	10/2023

Tabla 1: Comparación entre los modelos Gemini 1.5 Flash y GPT-4o Mini utilizados en este proyecto.

Además estos modelos, y sus variantes, son los que mejor resultados presentan en el *leaderbord*[4] de BIRD.

[3]Datos obtenidos de LLM Stats.

[4]BIRD-Bench Leaderboard ingresado el 11-11-2024.

3.4 Prompts

Lograr resultados consistentes en tareas Text-to-SQL con LLMs requiere una estructura estandarizada de *prompts* para permitir comparaciones justas entre modelos (Yang et al., 2024). El *prompt* consta de cuatro elementos: una instrucción, una base de datos, una pregunta en lenguaje natural y demostraciones adicionales (Chang and Fosler-Lussier, 2023).

La instrucción está diseñada para indicar de manera concisa la tarea a realizar a partir de la base de datos y la pregunta proporcionada.

Para representar la base de datos, se utilizo el método CREATE TABLE asegurando que las *keywords* y el esquema se presenten de manera uniforme (Yang et al., 2024).

La pregunta, por su parte, se utiliza directamente como se encuentra en el *dataset*.

Por último, se incluye conocimiento externo o evidencia adicional antes de la pregunta, con el fin de proporcionar contexto relevante (Yang et al., 2024). Los detalles de los *prompts* utilizados se encuentran en el apéndice C.

3.4.1 Zero-Shot

Las entradas incluyen la tarea y la pregunta con su correspondiente base de datos. El método se utiliza para evaluar directamente la capacidad de los modelos en tareas Text-to-SQL (Rajkumar et al., 2022) (Chang et al., 2023) (Liu et al., 2023).

Para este caso solo se le brindo el esquema de la base de datos, seguido de la pregunta. La siguiente variante es agregar conocimiento externo o evidencia que de más contexto a la pregunta realizada en relación a como esta formada la base de datos. (Zhang et al., 2024) (Chang and Fosler-Lussier, 2023).

3.4.2 Few-Shot

A partir de *prompt* anterior, se brindan tres ejemplos de preguntas, correspondientes a cada nivel de dificultad, junto con la *query* SQL que se espera. El objetivo es medir qué tan bien pueden los modelos realizar la tarea Text-to-SQL con un entrenamiento mínimo en datos del dominio específico (Chang and Fosler-Lussier, 2023) (Rajkumar et al., 2022).

4 Resultados

La Tabla 2 muestra los resultados de las métricas evaluadas, separadas en *Zero-shot* y *Few-shot*.

Se observa una clara mejora al darle más ejemplos al modelo para generar la *query* (*Few-shot*). Debido a que la métrica VES parte de los resultados de EX (Li et al., 2024), pero multiplicado por una constante que representa el tiempo de ejecución, presenta un valor mayor indicando que las *querys* generadas son eficientemente peor al ejecutarse comparadas con las *querys* verdades. La métrica que mejor resultados presento fue VA. Esta métrica no es de gran ayuda para el objetivo de este proyecto pero si para verificar que los modelos tienen la capacidad de generar *querys* que se puede ejecutar sin errores. Por ultimo la métrica CM es peor en cuanto a resultados, esto se debe a que compara uno a uno los resultados con las *querys* verdaderas.

Prompt	EX	VES	VA	CM
Zero-shot				
Gemini 1.5 Flash	55.8	56.9	97.7	72.5
GPT-4o Mini	51.2	51.8	99.2	71.8
Zero-shot con evidencia				
Gemini 1.5 Flash	73.6	74.7	90.7	72.4
GPT-4o Mini	81.4	83.0	100.0	73.6
Few-shot				
Gemini 1.5 Flash	68.2	67.8	97.7	76.9
GPT-4o Mini	60.5	61.5	96.9	74.4
Few-shot con evidencia				
Gemini 1.5 Flash	82.2	81.2	100.0	79.6
GPT-4o Mini	67.4	65.0	99.2	76.5

Tabla 2: Resultados de los modelos variando el *prompt* utilizando las métricas Execution Accuracy (EX), Valid Efficiency Score (VES), Valid SQL (VA) y Component Matching (CM) sobre la base de datos *Superhero* del *dataset* BIRD.

La figura 2 muestra los resultados agrupados según la dificultad de la pregunta. Se observa una como la métrica EX disminuye a medida que la pregunta se vuelve más compleja.

5 Conclusiones

Los resultados presentados en la Tabla 2 indican una mejora significativa en el desempeño de los modelos al utilizar ejemplos adicionales en la instrucción proporcionada. Esta tendencia sugiere que proporcionar contexto adicional, ya sea mediante ejemplos o más contexto, facilita la generación de *querys* más alineadas con los resultados esperados.

En cuanto a las métricas evaluadas, VES refleja un desempeño inferior en términos de eficiencia de rendimiento referidos al ejecutar la *query* en la base de datos, ya que parte de los resultados de EX pero escalados por una constante, lo que sugiere que las *querys* generadas requieren mayor tiempo de ejecución en comparación con las consultas de referencia.

Por otro lado, la métrica VA mostró los mejores resultados, mayores al 90 %, lo que confirma la capacidad de los modelo para generar consultas ejecutables sin errores sintácticos, aunque su utilidad en este estudio es limitada.

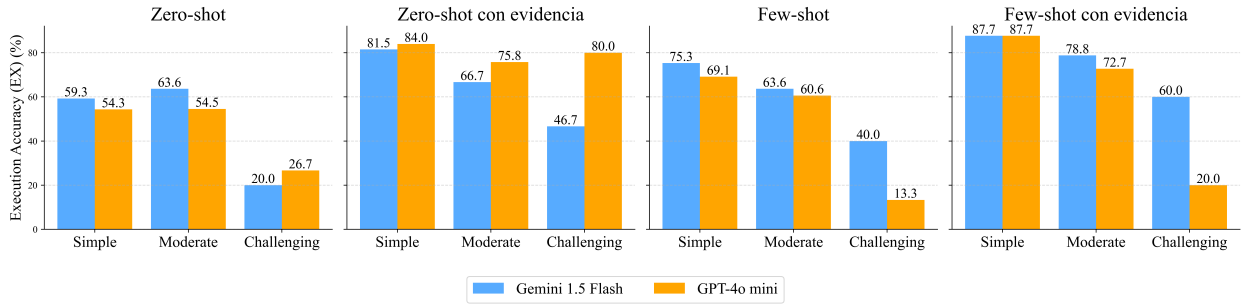

Figura 2: Resultados evaluando los modelos con Execution Accuracy (EX).

La métrica CM, en cambio, presentó los peores resultados, lo que era esperable debido a su enfoque de comparación exacta entre resultados, lo que la hace particularmente estricta.

Adicionalmente, al analizar los resultados en la Figura 2 se observa como los modelos tienden a disminuir su eficiencia en las *querys* generadas a medida que la complejidad de las preguntas aumenta, en especial en los niveles *Challenging*. Aunque para este último caso, GPT-4o mini tiene una amplia ventaja.

En general ambos modelos presentan resultados similares debido a que son versiones de los productos que compiten directamente. Esta batalla la podría haber ganado Gemini 1.5 Flash, si el usuario utiliza un instrucción y pregunta clara. Cuando menos contexto tiene, GPT-4o mini es superior.

Estos hallazgos resaltan la importancia de ajustar el número de ejemplos proporcionados y considerar estrategias que mejoren la precisión de las consultas generadas, especialmente en escenarios con preguntas de mayor dificultad.

6 Limitaciones

Este estudio proporciona resultados concretos a la hora de evaluar modelos en este tipo de tareas Text-to-SQL. El análisis fue usando el *dataset* de BIRD en el dominio de superhéroes, esta generalización esta limitada. Solo se exploro una pequeña base de datos para validar este trabajo, lo cual representa como se desempeñaba los modelos en una parte de todos los dominios.

Adicionalmente, el poder de computo utilizado fue limitado y debido a esto hubo que reducir la cantidad de modelos evaluados y utilizar una sola base de datos. Con esta limitación fue que se decidió utilizar solo dos modelos sobre una sola base de datos.

La elección de los modelos y solo dos tipos de *prompts* también es una posible limitación. Solo se emplearon dos modelos, Gemini 1.5 Flash y GPT-4o mini, y dos tipos diferentes de *prompts*. Evaluar más modelos y otras técnicas de *prompts* haber dado una visión más completa de cómo varían los resultados cambiando algún parámetro.

7 Trabajo Futuro

Para mejorar los resultados obtenidos, se proponen varias líneas de trabajo futuro. En primer lugar, la optimización del *prompt* mediante la exploración de técnicas avanzadas de selección de ejemplos permitirá mejorar la efectividad del enfoque *Few-shot*. Además, la evaluación de modelos más avanzados facilitará la comparación del desempeño con arquitecturas más recientes o especializadas en la generación de *querys* SQL (Zhu et al., 2024), lo que podría aportar mejoras significativas en términos de precisión y eficiencia.

Otro aspecto clave es el ajuste de las métricas, incorporando indicadores que no solo evalúen la exactitud de la *query* generada, sino también su eficiencia computacional en escenarios prácticos (Ma et al., 2024). Asimismo, la ampliación de la base de datos permitirá evaluar el desempeño del modelo en bases de datos más diversas y representativas, lo que contribuirá a una mejor generalización de los resultados.

Por último, la implementación de los modelos en un sistema real permitirá medir su impacto en tareas concretas de generación y optimización de consultas, proporcionando una evaluación más precisa de su aplicabilidad. Estas mejoras permitirán avanzar en la capacidad de los modelos para generar consultas más precisas, eficientes y aplicables a entornos reales.

Referencias

Shuaichen Chang and Eric Fosler-Lussier. 2023. How to prompt llms for text-to-sql: A study in zero-shot, single-domain, and cross-domain settings.

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. 2023. Dr.spider: A diagnostic evaluation benchmark towards text-to-sql robustness.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S. Yu. 2023. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability.

Xiping Liu and Zhao Tan. 2023. Divide and prompt: Chain of thought prompting for text-to-sql.

Limin Ma, Ken Pu, and Ying Zhu. 2024. Evaluating llms for text-to-sql generation with complex sql workload.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. 2024. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql.

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-sql capabilities of large language models.

Niklas Wretblad, Fredrik Gordh Riseby, Rahul Biswas, Amin Ahmadi, and Oskar Holmström. 2024. Understanding the effects of noise in text-to-sql: An examination of the bird-bench benchmark.

Jiaxi Yang, Binyuan Hui, Min Yang, Jian Yang, Junyang Lin, and Chang Zhou. 2024. Synthesizing text-to-sql data from weak and strong llms. *arXiv preprint arXiv:2408.03256*. 12 pages, 7 figures, ACL 2024.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *EMNLP*.

Bin Zhang, Yuxiao Ye, Guoqing Du, Xiaoru Hu, Zhishuai Li, Sun Yang, Chi Harold Liu, Rui Zhao, Ziyue Li, and Hangyu Mao. 2024. Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation. *arXiv preprint arXiv:2403.02951*. 26 pages, 6 figures, 14 tables.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suite. In *The 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Xiaohu Zhu, Qian Li, Lizhen Cui, and Yongkang Liu. 2024. Large language model enhanced text-to-sql generation: A survey.

A Base de datos

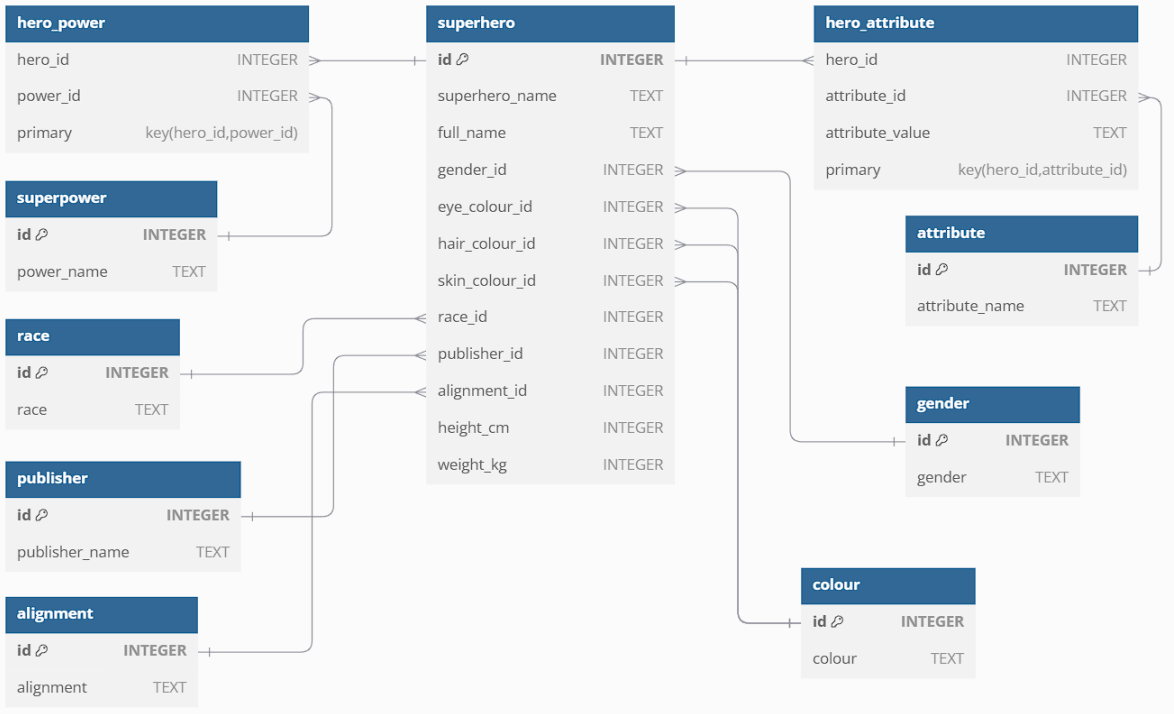

Figura 3: Esquema de la base de datos en el dominio superhéroes de BIRD.

La Figura 3 muestra el esquema de la base de datos utilizada. Contiene diversas tablas relacionadas al superpoder, atributos, editorial, atributos, genero entre otros, y todas orientadas a los superheroes de diversos comics. La descripción de cada tabla se encuentra en la Tabla 3.

Tabla	Descripción
superhero	Contiene la información básica de los superhéroes
hero_power	Habilidades especificas
superpower	Contiene los superpoderes
race	Raza a la que pertenece según sus características físicas
publisher	Editorial a la que pertenece
alignment	Alineación ética y moral
hero_atribute	Valor de los atributos
attribute	Lista de atributos
gender	Genero
colour	Color de piel, ojos, cabello, etc.

Tabla 3: Descripción de las tablas de la base de datos *superheros* de BIRD.

La base de datos consiste en 31 columnas distribuidas en 10 tablas. Aunque la mayoría de los nombres de las columnas son comprensibles, algunos plantean dificultades de interpretación, tal como se muestra en el esquema. En la práctica, un esquema de base de datos suele ir acompañado de un documento que explica en detalle cada tabla y columna. Esta documentación proporciona el contexto necesario para comprender el significado de cada elemento del esquema, el rango de valores posibles para campos con tipos no especificados y la lógica detrás de las relaciones. Sin esta documentación adicional, interpretar y utilizar eficazmente la base de datos puede resultar complicado. El conjunto de datos BIRD incluye una característica única para cada pregunta, denominada *hint*. Esta funcionalidad está diseñada para ofrecer información adicional o aclaraciones relacionadas con los detalles presentes en dicha documentación de la base de datos. Durante los experimentos, esta característica se proporciona a los *prompt* de los modelos descritos para cada pregunta.

B Código del esquema

En el Listing 1 se muestra el código para crear la base de datos. Este código es proporcionado en el *prompt*.

Listing 1: Esquema de la base de datos *superheros*

```
CREATE TABLE alignment (id INTEGER PRIMARY KEY, alignment TEXT);
CREATE TABLE attribute (id INTEGER PRIMARY KEY, attribute_name TEXT);
CREATE TABLE colour (id INTEGER PRIMARY KEY, colour TEXT);
CREATE TABLE gender (id INTEGER PRIMARY KEY, gender TEXT);
CREATE TABLE publisher (id INTEGER PRIMARY KEY, publisher_name TEXT);
CREATE TABLE race (id INTEGER PRIMARY KEY, race TEXT);
CREATE TABLE superpower (id INTEGER PRIMARY KEY, power_name TEXT);
CREATE TABLE superhero (
    id INTEGER PRIMARY KEY,
    superhero_name TEXT,
    full_name TEXT,
    gender_id INTEGER,
    eye_colour_id INTEGER,
    hair_colour_id INTEGER,
    skin_colour_id INTEGER,
    race_id INTEGER,
    publisher_id INTEGER,
    alignment_id INTEGER,
    height_cm INTEGER,
    weight_kg INTEGER,
    FOREIGN KEY (gender_id) REFERENCES gender(id),
    FOREIGN KEY (eye_colour_id) REFERENCES colour(id),
    FOREIGN KEY (hair_colour_id) REFERENCES colour(id),
    FOREIGN KEY (skin_colour_id) REFERENCES colour(id),
    FOREIGN KEY (race_id) REFERENCES race(id),
    FOREIGN KEY (publisher_id) REFERENCES publisher(id),
    FOREIGN KEY (alignment_id) REFERENCES alignment(id));
CREATE TABLE hero_attribute (
    hero_id INTEGER,
    attribute_id INTEGER,
    attribute_value TEXT,
    PRIMARY KEY (hero_id, attribute_id),
    FOREIGN KEY (hero_id) REFERENCES superhero(id),
    FOREIGN KEY (attribute_id) REFERENCES attribute(id));
CREATE TABLE hero_power (
    hero_id INTEGER,
    power_id INTEGER,
    PRIMARY KEY (hero_id, power_id),
    FOREIGN KEY (hero_id) REFERENCES superhero(id),
    FOREIGN KEY (power_id) REFERENCES superpower(id);
```

C Prompt

El *prompt* proporcionado para *zero-shot* y *few-shot* se muestra en los Listing 2 y 3 respectivamente. El *prompt* integra la instrucción, el esquema de la base de datos del apéndice B , la pregunta y, si corresponde, información adicional, para el que el modelo genere una *query* SQL. La información adicional brindada está diseñada para ofrecer información o datos complementarios necesarios para interpretar con precisión el esquema de la base de datos y para convertir correctamente la pregunta en una *query* SQL.

Listing 2: Zero-shot con evidencia

```
Given the database schema and question, generate ONLY a SQL query
SCHEMA: {database_schema}
QUESTION: How many superheroes have the super power of "Super Strength"?
HINT: super power of "Super Strength" refers to power_name = 'Super Strength'
```

Listing 3: Few-shot con evidencia

```
Given the database schema and question, generate ONLY a SQL query
SCHEMA: {database_schema}

Here are three example questions with increasing complexity and their corresponding SQL
     queries:

How many superheroes are there of each gender?
SELECT g.gender, COUNT(s.id) as total_heroes FROM superhero s JOIN gender g ON s.
    gender_id = g.id GROUP BY g.gender ORDER BY total_heroes DESC;

What is the average weight and height of superheroes by publisher, but only for
    publishers that have more than 5 superheroes?
SELECT p.publisher_name, ROUND(AVG(s.height_cm), 2) as avg_height, ROUND(AVG(s.
    weight_kg), 2) as avg_weight, COUNT(s.id) as total_heroes FROM superhero s JOIN
    publisher p ON s.publisher_id = p.id GROUP BY p.publisher_name HAVING COUNT(s.id) >
     5 ORDER BY total_heroes DESC;

What are the 5 most common superpowers among good-aligned heroes, including the
    percentage of good heroes that have each power?
WITH good_heroes AS (SELECT COUNT(DISTINCT s.id) as total_good_heroes FROM superhero s
    JOIN alignment a ON s.alignment_id = a.id WHERE a.alignment = 'Good') SELECT sp.
    power_name, COUNT(hp.hero_id) as heroes_with_power, ROUND(COUNT(hp.hero_id) * 100.0
     / gh.total_good_heroes, 2) as percentage_of_good_heroes FROM superpower sp JOIN
    hero_power hp ON sp.id = hp.power_id JOIN superhero s ON hp.hero_id = s.id JOIN
    alignment a ON s.alignment_id = a.id CROSS JOIN good_heroes gh WHERE a.alignment =
    'Good' GROUP BY sp.power_name, gh.total_good_heroes ORDER BY heroes_with_power DESC
     LIMIT 5;

QUESTION: Please list the full names of all the superheroes with over 15 super powers.
HINT: 15 super powers refers to COUNT(full_name) > 15
```