BACHELOR'S THESIS / ENGINEERING 2025

Del Lenguaje Natural a las Consultas SQL: Cómo los Modelos de Lenguaje Aprenden a Hablar con las Bases de Datos

Marc Bacaicoa Pau

upf. Universitat Pompeu Fabra Barcelona

Del Lenguaje Natural a las Consultas SQL: Cómo los Modelos de Lenguaje Aprenden a Hablar con las Bases de Datos

TRABAJO FIN DE GRADO DE

Marc Bacaicoa Pau

Director: Horacio Saggion

Grado en Ingeniería en Informática

Curso 2024-2025

upf. Universitat Pompeu Fabra Barcelona

Escola d'Enginyeria

Agradecimientos

Este trabajo es más que un conjunto de datos y gráficas; es el resultado de una experiencia que no podría haber llevado a cabo solo. Estas líneas están dedicadas a aquellos que han sido mi apoyo, mi motivación y mi soporte constante.

A Horacio Saggion, gracias por tu apoyo y tu paciencia durante todo el proceso. Tus conocimientos y consejos han sido de gran ayuda para conseguir obtener un resultado mejor del que pensaba.

A mis padres, quienes han estado a mi lado desde mi primer día de vida y no han dudado de mis capacidades. Gracias por estar aun cuando era complicado hacerlo.

A mis abuelos y mi tía, por ser un pilar fundamental a lo largo de toda mi educación.

Y a Begoña, mi compañera de vida, gracias por estar a mi lado y mostrar cariño y paciencia. No podría haber hecho esto sin ti.

Y, por último, gracias a todos los investigadores cuyos trabajos y proyectos forman parte de los fundamentos de este trabajo. Vuestra dedicación en el avance del conocimiento ha hecho posible este trabajo.

Resumen

Este trabajo explora cómo los modelos de lenguaje son capaces de traducir preguntas en lenguaje natural a consultas SQL, comparando dos arquitecturas distintas: un modelo encoder-decoder (T5-small-finetuned-WikiSQL) y uno decoder-only (Gemma-2b). Para ello, se ha utilizado el dataset Spider, que presenta una alta complejidad estructural y gramatical. Ambos modelos se evaluaron antes y después de aplicar fine-tuning, y se analizó su rendimiento según el tipo de consulta y la dificultad del lenguaje de entrada. Los resultados muestran que T5, a pesar de su menor tamaño, supera a Gemma en precisión tras el fine-tuning. Esto sugiere que la arquitectura y la especialización previa son factores más determinantes que el tamaño del modelo. Este trabajo concluye que adaptar los modelos al dominio SQL es clave para mejorar su rendimiento en tareas estructuradas.

Resum

Aquest treball analitza com els models de llenguatge poden traduir preguntes en llenguatge natural a consultes SQL, comparant dues arquitectures diferents: un model encoder-decoder (T5-small-finetuned-WikiSQL) i un altre decoder-only (Gemma-2b). Per fer-ho, s'ha utilitzat el conjunt de dades Spider, conegut per la seva alta complexitat. Tots dos models han estat avaluats abans i després de l'aplicació del fine-tuning, examinant el seu rendiment segons el tipus de consulta i la dificultat lingüística. Els resultats mostren que el model T5, malgrat ser més petit, obté millors resultats després de l'ajustament, fet que destaca la importància de l'arquitectura i de l'especialització prèvia per sobre de la mida. El treball conclou que l'adaptació al domini SQL és fonamental per millorar el rendiment en tasques estructurades.

Abstract

This project explores how language models can translate natural language questions into SQL queries, comparing two different architectures: an encoder-decoder model (T5-small-finetuned-WikiSQL) and a decoder-only model (Gemma-2b). The Spider dataset, known for its structural complexity, was used for both evaluation and fine-tuning. Each model was tested before and after fine-tuning, and their performance was analyzed by query type and input linguistic complexity. Results show that T5-small, despite being smaller, outperformed Gemma after fine-tuning, highlighting that model architecture and prior specialization are more relevant than size alone. The study concludes that adapting models to the SQL domain is essential to improve performance in structured tasks

Índice

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

1.1 Contexto

Este trabajo se enmarca en el campo de los modelos de lenguaje de gran escala (Large Language Models - LLMs), una categoría de modelos de procesamiento de lenguaje natural (NLP) que han experimentado un crecimiento significativo a lo largo de estos últimos años (Naveed et al., 2023). Aunque los modelos de lenguaje tienen sus raíces en técnicas estadísticas básicas como los modelos de n-gramas, cuyas aplicaciones modernas fueron ampliamente desarrolladas por P. F. Brown et al. (1992), el término LLM se asocia principalmente a la introducción de modelos basados en arquitecturas profundas y preentrenamiento masivo sobre grandes volúmenes de texto (Brown et al., 2020).

Antes de la aparición de los LLMs, los modelos de lenguaje evolucionaron desde enfoques puramente estadísticos hasta arquitecturas neuronales más avanzadas. Inicialmente se emplearon los modelos de n-gramas, formalizados desde mediados del siglo XX y posteriormente refinados por P. F. Brown et al. (1992), que estiman la probabilidad de la próxima palabra basándose únicamente en un número fijo de palabras anteriores.

A comienzos de la década de 2000 se introdujeron las redes neuronales recurrentes (RNN), demostrando reducciones significativas de *perplexity*, una métrica ampliamente utilizada que refleja la capacidad del modelo para asignar probabilidades bajas a palabras inesperadas y altas a las esperadas dentro de un chunk de texto (Mikolov et al., 2010). Esta métrica cuantifica la incertidumbre del modelo al predecir la siguiente palabra y ha sido objeto de revisión crítica en trabajos recientes como Meister y Cotterell (2021) en los que se discuten sus fortalezas y limitaciones frente a métodos más tradicionales como los enfoques back-off, que recurren a n-gramas de menor longitud cuando no se encuentran coincidencias suficientes en los más largos.

No obstante, las RNN tradicionales sufrían dificultades para aprender dependencias a largo plazo debido al problema del desvanecimiento de gradiente durante el entrenamiento (Bengio et al., 1994). Para mitigar este problema, Hochreiter y Schmidhuber (1997) propusieron las Long Short-Term Memory (LSTM), una arquitectura que introduce una celda de memoria controlada por mecanismos de puerta, permitiendo preservar y actualizar información relevante a lo largo de secuencias extensas sin degradación del gradiente.

Paralelamente, Collobert et al. (2011) propusieron redes convolucionales aplicadas a NLP, ofreciendo arquitecturas unificadas para el aprendizaje de representaciones sin necesidad de ingeniería manual de características. Estas innovaciones sentaron las bases para la arquitectura Transformer de Vaswani et al. (2017), que introdujo el mecanismo de atención como alternativa más eficiente a las redes recurrentes y convolucionales, capturando dependencias a largo plazo y marcando un claro punto de inflexión en la evolución de los LLMs.

Posteriormente, Devlin et al. (2018) presentaron BERT (Bidirectional Encoder Representations from Transformers), un modelo basado en Transformers bidireccionales pre-entrenado sobre grandes cantidades de texto sin etiquetar, marcando así una tendencia clara hacia modelos de lenguaje más potentes y flexibles capaces de capturar mejor el contexto semántico y sintáctico del lenguaje natural.

En ese mismo año, Radford et al. (2018) propusieron el modelo GPT (Generative Pre-trained Transformer**)**, una arquitectura exclusivamente basada en el bloque *decoder* del Transformer, orientada a la generación de texto coherente y realista. Este modelo y sus sucesivas versiones GPT-2 (Radford et al., 2019**),** GPT-3 (T. B. Brown et al., 2020) y GPT-4 (OpenAI, 2023), han establecido nuevos límites

en capacidades de generación y comprensión lingüística, alcanzando resultados cercanos al desempeño humano en tareas como comprensión de texto, generación de código, resolución de instrucciones y QA compleja (T. B. Brown et al., 2020; OpenAI, 2023).
Actualmente, los LLMs se aplican en múltiples dominios, y, tal y como demuestran Zhu et al. (2024), uno de los más prometedores es la traducción automática de lenguaje natural a consultas SQL, conocida como *text-to-SQL.*

Uno de los primeros modelos pioneros en este campo fue Seq2SQL, propuesto por Zhong et al. (2017), que introdujo el uso de modelos de aprendizaje profundo para mapear instrucciones en lenguaje natural a sentencias SQL. Esta tarea busca facilitar la interacción con bases de datos mediante instrucciones en lenguaje natural, permitiendo que usuarios sin conocimientos técnicos generen consultas SQL válidas.

Gracias a su capacidad de generalización y comprensión semántica, los LLMs pueden interpretar enunciados complejos, adaptarse a distintos esquemas y generar sentencias estructuradas de forma autónoma. Esta línea de trabajo representa una evolución clave hacia interfaces más accesibles e inteligentes en el ámbito del análisis de datos y la explotación de información.

En este trabajo nos preguntamos cuáles son las capacidades reales de estos modelos para comprender instrucciones expresadas en lenguaje humano y transformarlas en consultas estructuradas sobre bases de datos, es decir, consultas en lenguaje SQL. Esta investigación resulta clave em el desarrollo de sistemas que permitan que los usuarios interactuar de manera intuitiva con grandes volúmenes de datos sin necesidad de conocimientos técnicos avanzados.

1.2 Objetivos generales y específicos

Este proyecto centra su enfoque en evaluar de forma comparativa el rendimiento de dos LLMs, uno basado en una arquitectura encoder-decoder y otro basado en una arquitectura decoder-only, en la tarea de generación automática de consultas SQL a partir de lenguaje natural. Esta tarea representa un reto significativo en el ámbito del NLP, ya que requiere no solo comprender la intención semántica del enunciado, sino también traducirla a una estructura formal y ejecutable sobre bases de datos relacionales.

Para ello, se usa como base de datos de evaluación y para realizar fine-tuning, el conjunto Spider, un conjunto de datos propuesto por Yu et al. (2018), y que son ampliamente utilizados en benchmarks del campo text-to-SQL. Spider presenta una gran diversidad de esquemas de bases de datos y consultas de distinta complejidad, lo que lo convierte en un benchmark modelo para medir la capacidad de los modelos de lenguaje en esta tarea.

Diversos trabajos se han interesado en esta cuestión, proponiendo arquitecturas y estrategias de entrenamiento específicas para mejorar el rendimiento de este benchmark, entre ellos, destaca IRNet, propuesto por Guo et al. (2019), que introduce una representación semántica intermedia antes de generar la consulta SQL o RAT-SQL de Wang et al. (2020), que incorpora mecanismos de atención relacional para modelar la estructura del esquema de la base de datos.

Este análisis se centra en determinar cómo influye la arquitectura interna de cada modelo en la precisión de las consultas generadas, teniendo en cuenta la complejidad tanto de la entrada en lenguaje natural como de la salida, la consulta en lenguaje SQL.

A partir de este planteamiento general, se derivan los siguientes objetivos específicos:

- Revisar y analizar el estado del arte en generación de consultas SQL a partir de lenguaje natural, con especial atención a los modelos basados en transformadores y su aplicación en datasets como Spider y WikiSQL.

- Diseñar y ejecutar un proceso de fine-tuning eficaz para ambos modelos utilizando el dataset Spider, optimizando su rendimiento en la generación de consultas SQL.
- Analizar la influencia de la longitud y la complejidad léxica de los enunciados en lenguaje natural sobre la calidad de las consultas generadas, considerando los resultados previos y posteriores al ajuste.
- Evaluar la similitud entre las consultas generadas y las de referencia mediante la métrica BLEU, propuesta por Papineni et al. (2001), tanto antes como después del fine-tuning.
- Identificar y discutir las ventajas y limitaciones de cada arquitectura a partir del análisis cuantitativo y cualitativo de los resultados obtenidos en todo el proceso experimental.

1.3 Estructura del documento

Este Trabajo de Fin de Grado se organiza en seis capítulos, cada uno de los cuales aborda un aspecto que ha sido clave durante el desarrollo del estudio:

- **Capítulo 1 - Introducción**: Contextualización del problema y presentación de los objetivos generales y específicos del trabajo.
- **Capítulo 2 - Estado del Arte y Marco Teórico**: Descripción de los fundamentos del procesamiento del lenguaje natural y LLMs, así como el campo de investigación *text-to-SQL*. También se presentan los datasets utilizados y los distintos tipos de consultas SQL considerados en el estudio.
- **Capítulo 3 - Metodología**: detallado de los modelos seleccionados, las características de los datos empleados, el proceso experimental llevado a cabo y los criterios de evaluación utilizados.
- **Capítulo 4 - Resultados**: se presentan los resultados obtenidos en la evaluación de los modelos, clasificados por tipo de consulta y por nivel de complejidad léxica del input en lenguaje natural.
- **Capítulo 5 - Discusión**: se analizan de forma crítica los resultados obtenidos, comparando el rendimiento de los modelos y reflexionando sobre los factores que afectan a su desempeño.
- **Capítulo 6 - Conclusiones y Trabajo Futuro**: resumen de los principales hallazgos del estudio, así como desatacar las contribuciones del trabajo y propuestas de posibles líneas de investigación futura.

Capítulo 2

Estado del Arte y Marco Teórico

2.1 Procesamiento del lenguaje natural y su relación con SQL

El procesamiento del lenguaje natural (Natural Language Processing, NLP) es una rama de la inteligencia artificial centrada en la interacción entre seres humanos y máquinas a través del lenguaje, tanto hablado como escrito (Jurafsky & Martin, 2010).

Una de las áreas emergentes dentro del NLP es la traducción del lenguaje natural a lenguajes formales, como el lenguaje de consulta estructurado (Structured Query Language, SQL). Esta tarea, conocida como text-to-SQL, tiene como objetivo generar automáticamente consultas SQL a partir de enunciados en lenguaje natural, como propusieron Yin y Neubig (2017), mediante un modelo sintáctico que respeta la estructura del lenguaje objetivo. Asimismo, enfoques como el de Elgohary et al. (2020) introducen la interacción del usuario para facilitar el proceso de formulación de consultas.

La relación entre NLP y SQL se muestra, por tanto, en la necesidad de interpretar instrucciones expresadas en lenguaje natural y transformarlas en estructuras formales que cumplan con las reglas sintácticas de un sistema de gestión de bases de datos (DBMS) (Kim et al., 2020). Esta traducción automática plantea retos significativos tanto desde la perspectiva lingüística, abarcando temas como la ambigüedad, la polisemia o las construcciones compuestas, como desde la dimensión estructural, que exige referenciar correctamente tablas, atributos, condiciones y relaciones entre entidades (Yu et al., 2018).

2.2 Modelos de lenguaje de gran escala

Los modelos de lenguaje de gran escala han transformado el campo del procesamiento de lenguaje natural en los últimos años. Estos modelos se caracterizan por contar miles de millones de parámetros y haber sido entrenados sobre enormes cantidades de datos textuales, lo que permite capturar patrones lingüísticos complejos, inferencias semánticas y relaciones sintácticas a largo plazo (Naveed et al., 2023).

Su arquitectura se basa normalmente en el mecanismo de atención introducido por Transformers, lo que permite procesar secuencias de texto de forma paralela y capturar dependencias contextuales a largo plazo con mayor eficiencia que los enfoques tradicionales basados en redes recurrentes o convolucionales. Gracias a este avance, modelos como BERT y GPT han demostrado capacidades sin precedentes en tareas como clasificación de texto, resumen automático, generación de lenguaje y, de forma más reciente, traducción de lenguaje natural a lenguajes estructurados como SQL.

Dentro del conjunto de LLMs, se pueden distinguir dos enfoques de arquitectura predominantes: los modelos encoder-decoder, y los modelos decoder-only.

2.2.1 Arquitecturas Encoder-Decoder

Los modelos encoder-decoder, inicialmente propuestos para tareas de traducción automática (Sutskever et al., 2014), constan de dos componentes principales. Tal y como se muestra en la Figura 1, constan de: un codificador (encoder), que convierte el input (texto en lenguaje natural) en una representación interna o vector de contexto, y un decodificador (decoder), que genera la secuencia de salida (en este caso, una consulta SQL) a partir de dicha representación.

Si hablamos de tareas de secuencia a secuencia (seq2seq), esta arquitectura resulta especialmente útil, sobre todo en tareas como la traducción de idiomas, el resumen automático o la generación de consultas estructuradas. Un ejemplo de este enfoque es el modelo T5 (Text-To-Text Transfer Tansformer), que reformula todas las tareas de NLP como problemas de traducción entre texto de entrada y texto de salida (Raffel et al., 2019). Gracias a su entrenamiento multitarea, T5 ofrece una adaptación flexible a tareas como text-to-SQL, al permitir la generación controlada de salidas estructuradas a partir de lenguaje natural.

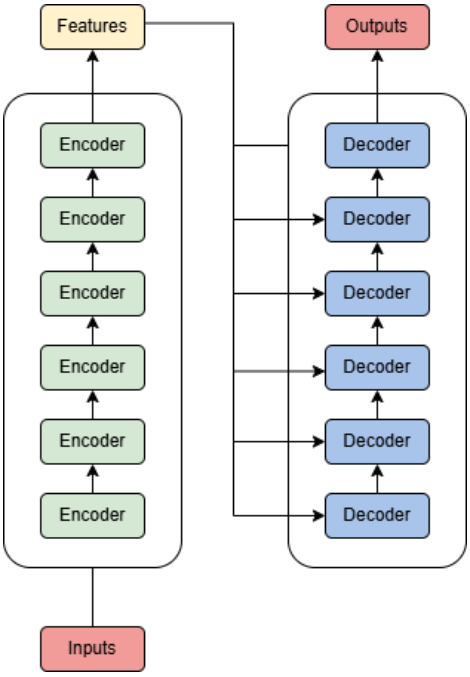


Figura 1: Diagrama propio simplificado de arquitectura encoder-decoder.

2.2.2 Arquitecturas Decoder-Only

Por otro lado, los modelos decoder-only, como la familia GPT, eliminan la fase de codificación y emplean únicamente un decodificador autorregresivo para generar texto palabra a palabra, condicionado por el contexto previo (Radford et al., 2019). Esta arquitectura ha demostrado una gran capacidad para generar texto coherente y adaptado a una amplia variedad de tareas sin requerir entrenamiento adicional, los llamados zero-shot, few-shot o prompt-based learning (T. B. Brown et al., 2020).

En el caso de las tareas text-to-SQL, estos modelos se alimentan de una instrucción en lenguaje natural y generan directamente la consulta SQL como respuesta. En la Figura 2, se pude observar que únicamente constan de un único componente, el decodificador. Si bien su capacidad de generalización es notable, pueden verse limitados en escenarios donde es importante modelar ciertas relaciones estructurales complejas entre entidades de base de datos, como se evidencia en estudios recientes que proponen mecanismos de validación incremental para corregir estos defectos (Scholak et al., 2021).

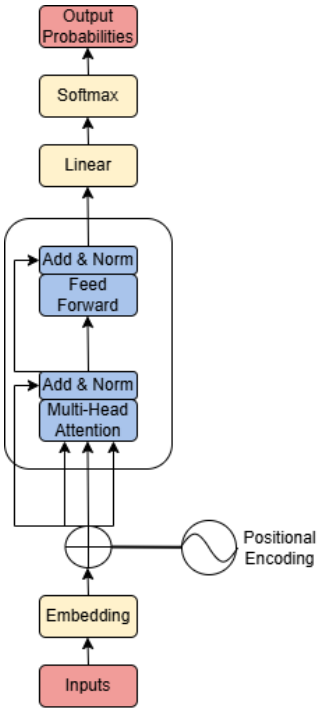


Figura 2: Diagrama propio de arquitectura Decoder-only.

2.3 La tarea Text-to-SQL: definición, retos y aplicaciones

La tarea conocida como text-to-SQL consiste en traducir automáticamente instrucciones en lenguaje natural a consultas estructuradas en lenguaje SQL, que pueden ser ejecutadas sobre bases de datos relacionales (Nascimento et al., 2024).

Desde el punto de vista computacional, se trata de un problema de generación de lenguaje condicionado, en el que el modelo debe comprender la intención semántica de la instrucción, identificar las entidades, operadores y filtros relevantes, y componer una consulta SQL funcional que sea válida de forma sintáctica. A diferencia de otras tareas clásicas de NLP, el problema text-to-SQL añade una capa estructural, ya que el modelo debe tener en cuenta tanto el esquema de la base de datos como las relaciones entre tablas y atributos. Esta complejidad adicional ha sido ampliamente discutida en trabajos como los de Yu et al. (2018) y Cao et al. (2021), que destacan la necesidad de modelar tanto la estructura de la base de datos como las restricciones lógicas para generar consultas SQL válidas y precisas.

Los principales desafíos de esta tarea incluyen:

- Ambigüedad lingüística: una misma consulta puede formularse de múltiples formas, y algunas expresiones son difíciles de interpretar sin contexto adicional (Bhaskar et al., 2023).
- Complejidad sintáctica: consultas con unión de 2 o más tablas (JOINs), condiciones añadidas, agrupaciones o subconsultas requieren una generación precisa y ordenada (Yu et al., 2018).
- Generalización entre dominios: los modelos deben ser capaces de funcionar sobre bases de datos no vistas durante el entrenamiento (cross-domain generalization) (Gan et al., 2021).
- Desalineación entre lenguaje natural y estructura de la base de datos: el vocabulario del usuario no siempre coincide con los nombres de columnas o tablas. (Gan et al., 2021).

Esto ha impulsado aplicaciones prácticas como asistentes conversacionales para bases de datos o herramientas de análisis exploratorio de datos accesibles (Rajkumar et al., 2022)

2.4 Benchmarks y datasets de referencia

Para evaluar de forma objetiva el rendimiento de los modelos en tareas de generación automática de consultas SQL a partir de lenguaje natural, es imprescindible contar con conjuntos de datos estandarizados que actúen como benchmarks de referencia. Estos benchmarks permiten comparar resultados bajo condiciones homogéneas y reproducibles, y son fundamentales tanto para la fase de entrenamiento como para la validación y prueba de los modelos.

2.4.1 Spider

Spider es uno de los benchmarks más exigentes y completos para evaluar tareas de generación automática de consultas SQL a partir de lenguaje natural. Fue propuesto por Yu et al. (2018) con el objetivo de medir la capacidad de generación semántica y estructural de los modelos text-to-SQL.

Este conjunto de datos contiene un total de 10.181 pares de pregunta-consulta SQL, distribuidos sobre 200 esquemas de bases de datos relacionales que cubren una gran variedad de dominios (educación, comercio, deportes, música, geografía, etc.). Estas bases de datos están diseñadas para incluir estructuras reales y complejas, con múltiples tablas relacionadas entre sí mediante claves foráneas (foreign keys), lo que permite generar consultas SQL no triviales.

Para la distribución final del conjunto de datos de entrenamiento, además de los 10.181 ejemplos iniciales, se han añadido 1659 pares que siguen el mismo protocolo de anotación de los siguientes datasets. Estos provienen de trabajos anteriores previamente utilizados, como los de Zelle & Mooney (1996) para GeoQuery, Li y Jagadish (2014) para Academic, Yaghmazadeh et al. (2017) para IMDB y Yelp, Iyer et al. (2017) para Scholar, y diversas fuentes para Restaurants (Tang & Mooney, 2000; Popescu et al., 2003; Giordani & Moschitti, 2012), quienes las reunieron con el objetivo de evaluar de forma más uniforme la generación de SQL a partir de lenguaje natural.

La distribución del dataset es la siguiente:

- Conjunto de entrenamiento: 8.659 ejemplos divididos en 146 bases de datos distintas.
- Conjunto de desarrollo (dev): 1.034 ejemplos divididos en 20 bases de datos distintas no presentes en el conjunto de entrenamiento.
- Conjunto de test (no público): 2.147 ejemplos divididos en 40 bases de datos nuevas y en las ya existentes en el conjunto de entrenamiento.

Esta separación entre bases de datos de entrenamiento y prueba convierte a Spider en un benchmark de evaluación tipo cross-domain generalization, es decir, los modelos no pueden memorizar estructuras específicas y deben aprender a adaptarse a esquemas de bases de datos que no han sido vistos previamente.

Dentro de Spider, cada ejemplo del dataset incluye:

- La pregunta en lenguaje natural.
- El esquema de la base de datos (nombres de tablas y columnas).
- La consulta SQL objetivo.
- La estructura de la consulta representada como árbol sintáctico (SQL query parse tree).

2.4.2 WikiSQL

WikiSQL es un benchmark propuesto por Zhong et al. (2017) con el objetivo de evaluar la capacidad de los modelos para traducir instrucciones en lenguaje natural a consultas SQL sobre tablas simples extraídas de Wikipedia. A diferencia de otros conjuntos más recientes y complejos, como podría ser Spider, WikiSQL se caracteriza por su estructura más controlada y limitada, lo que lo convierte en una opción adecuada para tareas de iniciación, evaluación rápida o entrenamiento preliminar de modelos.

El conjunto de datos de WikiSQL contiene un total de 80.654 ejemplos, divididos en entrenamiento (61.297 ejemplos), validación (9.145 ejemplos) y test (10.221 ejemplos).

Todas las consultas SQL están limitadas a una sola tabla (single-table queries), lo que impide realizar JOINs, subconsultas u otras consultas complejas. Las consultas generadas por los modelos deben basarse únicamente en las columnas que estén visibles en esa tabla concreta de la que se quiere extraer la consulta SQL.

Debido a estas limitaciones, WikiSQL se emplea habitualmente como complemento a otros benchmarks más complejos y exigentes, como puede ser Spider, o bien como dataset de preentrenamiento antes de aplicar fine-tuning sobre conjuntos más complejos.

2.5 Métricas de evaluación: BLEU

La evaluación de modelos de generación de texto requiere métricas que puedan cuantificar de forma objetiva la calidad del texto generado en comparación con una referencia humana. En este proyecto, se ha utilizado la métrica BLEU (Bilingual Evaluation Understudy), que fue propuesto por primera vez por Papineni et al. (2002) como método automático para evaluar sistemas de traducción automática.
Dentro de la comunidad de NLP, una de las primeras métricas ampliamente aceptadas fue BLEU, que hoy sigue siendo usada por su simplicidad y eficiencia computacional (Reiter E., 2018). Aunque fue desarrollada originalmente para traducción, su enfoque basado en la comparación de n-gramas permite su aplicación en tareas como generación de texto, resumen, y, como en este caso, traducción de lenguaje natural a consultas SQL.

Tal y como fue propuesto por Papineni et al. (2002), el método BLEU mide la coincidencia entre la salida generada por el modelo y una o más referencias humanas, considerando segmentos consecutivos de palabras, llamados n-gramas. A mayor coincidencia de n-gramas, mayor será el valor de BLEU.
El valor final de BLEU es una puntuación entre 0 y 1, donde 0 indica una coincidencia nula o irrelevante, y 1 indica una coincidencia perfecta.

El valor de la puntuación se calcula de la siguiente forma:

- Precisión de n-gramas: proporción de n-gramas generados que coinciden con la referencia.
- Penalización por longitud (brevity penalty): penaliza respuestas muy cortas que podrían coincidir de forma artificial con la referencia.

La fórmula general con la que se realiza el cálculo es:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} \omega_n \cdot \log p_n\right)$$

Donde:

- p_n: precisión de n-gramas de orden n
- ω_n: peso para cada orden n
- BP: brevity penalty

Aunque BLEU es una métrica estándar, existen diversas variantes según su configuración que permiten ajustar la evaluación según el nivel de detalle que se requiere. Estas variantes difieren principalmente en el orden de n-gramas que se considera, la aplicación de técnicas de suavizado (smoothing) y el nivel sobre el que se calcula la métrica, ya sea corpus completo o frase individual (Chen & Cherry, 2014).

A continuación, se presenta una tabla comparativa que resume las principales variantes de BLEU:

Tabla 1: Tabla comparativa propia de las principales variantes de BLEU.

	N-gramas utilizados	**Descripción**
BLEU-1	Unigramas (n=1)	Evalúa precisión palabra a palabra.
BLEU-2	Unigramas y bigramas	Añade relaciones entre pares de palabras.
BLEU-3	Hasta trigramas (n=3)	Mayor contextualización local.
BLEU-4	Hasta cuatrigramas (n=4)	Versión estándar más común.
Smoothed BLEU	n-gramas con suavizado	Añade correcciones para evitar puntuaciones 0.
Sentence-level BLEU	Evaluación por frase	Aplica BLEU a nivel de oración (en lugar de corpus).

Por otro lado, BLEU tiene ciertas limitaciones que impiden que sea una métrica perfecta, como, por ejemplo:

- No considera sinónimos ni equivalencias semánticas (Ren et al., 2020).
- Penaliza en exceso las diferencias superficiales (Reiter E., 2018).
- No evalúa la validez sintáctica ni lógica (especialmente relevante en SQL) (Ren et al., 2020).

Debido a estas limitaciones, y relacionado con la evolución de este campo, se puede complementar BLEU con otras métricas como:

- Exact Match (EM): evalúa si todos los componentes estructurales de la consulta generada coinciden exactamente con los de la consulta de referencia, permitiendo cierto margen en el orden de algunas cláusulas.
- Execution Accuracy (EX): comprueba si la consulta generada devuelve el mismo resultado que la consulta de referencia al ejecutarse sobre la base de datos, aunque su forma textual no coincida exactamente.

Ambas métricas han adquirido un papel central en la evaluación de tareas text-to-SQL, especialmente a partir de su adopción sistemática en el benchmark Spider (Yu et al., 2018), donde se establecieron como criterios fundamentales para medir la precisión estructural y funcional de las consultas generadas.
No obstante, BLEU sigue siendo útil como indicador cuantitativo inicial, particularmente en escenarios comparativos entre modelos.

Capítulo 3

Metodología

Este capítulo presenta la metodología empleada para analizar y evaluar comparativamente el rendimiento de dos modelos de lenguaje. El enlace a GitHub con el código desarrollado para este trabajo se incluye en el Apéndice A.1. El objetivo es determinar en qué medida estos modelos son capaces de generar consultas correctas sobre bases de datos relacionales complejas, utilizando como referencia el benchmark Spider.

3.1 Selección y descripción de los modelos

El objetivo principal, tal y como se detalla en el apartado 1.2 de este trabajo, es evaluar la capacidad de dos modelos de lenguaje para resolver tareas del tipo text-to-SQL, es decir, la generación automática de consultas SQL a partir de instrucciones formuladas en lenguaje natural.

Para abordar esta tarea, se han seleccionado dos modelos de lenguaje de código abierto, accesibles desde la plataforma Hugging Face (*Hugging Face – The AI Community Building The Future.*, s. f.), que presentan arquitecturas y niveles de especialización diferentes:

- **T5-small-finetuned-wikisql** (mrm8488, 2020), un modelo encoder-decoder basado en la arquitectura T5, previamente ajustado sobre el dataset WikiSQL, y optimizado para consultas SQL sobre esquemas simples.
- **Gemma-2b** (Google, 2024), un modelo decoder-only de aproximadamente 2.000 millones de parámetros, sin ajuste previo específico para tareas SQL, diseñado para la generación de texto general a través de prompting.

La elección de modelos con arquitecturas distintas permite analizar como influye el diseño interno del modelo en calidad de las consultas generadas. Asimismo, se consideran diferencias sustanciales en cuanto a tamaño, especialización previa y compatibilidad con tareas estructuradas como la generación de consultas SQL.

Ambos modelos se han evaluado bajo dos enfoques complementarios:

- **Inferencia directa (sin fine-tuning):** para medir su rendimiento base en la tarea sin entrenamiento adicional.
- **Fine-tuning sobre el dataset Spider:** para analizar hasta que punto son capaces de adaptarse a esquemas complejos mediante entrenamiento específico.

Esta comparación entre modos de uso permite identificar la influencia real del fine-tuning sobre la calidad de las predicciones, así como examinar la capacidad de generalización de los modelos a estructuras no vistas. Al evaluar modelos de distinta arquitectura y grado de especialización, se busca determinar qué combinación resulta más eficaz para la traducción de lenguaje natural a SQL, tanto en escenarios simples como en contextos estructuralmente más exigentes.

3.1.1 T5-small-finetuned-wikisql (Encoder-Decoder)

El primer modelo analizado es t5-small-finetuned-wikisql[1], publicado por mrm8488 (2020), una versión reducida del modelo T5 propuesto por Raffel et al. (2019). Este modelo utiliza una arquitectura encoder-decoder, diseñada para reformular cualquier tarea de procesamiento del lenguaje natural como una tarea de traducción entre el texto de entrada y el texto de salida. La versión *small* contiene aproximadamente 60 millones de parámetros, lo que facilita su uso en entornos con recursos limitados, como Google Colab.

Este modelo ha sido previamente entrenado y ajustado sobre el dataset WikiSQL, propuesto por Zhong et al. (2017), que como se describe en el apartado 2.4.2, está limitado a consultas sobre una única tabla. Por tanto, este modelo ha sido entrenado específicamente para generar SQL sobre esquemas simples y sin relaciones complejas, lo que condiciona su capacidad de generalización a escenarios como Spider.

Se ha elegido este modelo por su especialización previa en tareas text-to-SQL simples, lo que permite analizar hasta que punto puede adaptarse a esquemas más complejos como los que presenta el dataset Spider. En este trabajo, se han utilizado dos modalidades de evaluación complementarias:

- **Inferencia directa (Zero-shot):** Se evalúa el modelo tal y como está publicado en Hugging Face, sin ningún ajuste adicional, aplicando directamente la inferencia sobre el conjunto de validación de Spider. Esta modalidad permite establecer una línea base de su rendimiento fuera del dominio para el que fue entrenado.

- **Fine-tuning supervisado sobre Spider:** se reentrena el modelo utilizando el conjunto de entrenamiento de Spider, con el objetivo de especializarlo en estructuras SQL más complejas. Esta adaptación permite evaluar si el modelo puede aprender a generar consultas sobre bases de datos relacionales con múltiples tablas, claves foráneas y condiciones estructurales avanzadas.

Durante el entrenamiento y la inferencia, el modelo recibe las preguntas con el siguiente *prompt,* extraído directamente de la documentación del modelo:

```
"Translate English to SQL: {query} </s>"
```

Figura 3: Prompt utilizado por el modelo t5-small-finetuned-wikisql.

Dado el preentrenamiento específico sobre estructuras simples, se espera que el modelo tenga dificultades en su versión sin fine-tuning para manejar consultas que requieran combinaciones de tablas o lógica compleja. Sin embargo, se plantea que, tras el fine-tuning sobre Spider, el modelo pueda mejorar significativamente su rendimiento, aprovechando su capacidad encoder-decoder para aprender patrones más complejos y estructurados.

3.1.2 Gemma-2b (Decoder-Only)

El segundo modelo seleccionado es *gemma-2b*[2], un modelo perteneciente a la familia de modelos Gemma, desarrollada por Google Deepmind (Google DeepMind, 2024). Este modelo utiliza una arquitectura decoder-only, de tipo autoregresiva, similar a la empleada en los modelos GPT (Radford et

[1] https://huggingface.co/mrm8488/t5-small-finetuned-wikiSQL
[2] https://huggingface.co/google/gemma-2b

al., 2019; Brown et al., 2020), en la que la generación de texto se realiza de manera secuencial, condicionada únicamente por el contexto anterior.

La versión utilizada, gemma-2b, contiene aproximadamente 2.000 millones de parámetros, lo que proporciona una capacidad de representación mucho mayor en comparación con los modelos de menor tamaño como t5-small.

En este caso, el modelo no ha sido ajustado previamente para la tarea text-to-SQL, ni ha sido entrenado específicamente sobre bases de datos o estructuras SQL. En su fase de preentrenamiento, Gemma fue expuesto a grandes volúmenes de texto variado en distintos dominios, lo que le permite adquirir conocimientos generales del lenguaje natural, pero sin una comprensión explícita de sintaxis estructurada como la de SQL.

Este enfoque generalista plantea un desafío interesante: determinar si un modelo no especializado puede adaptarse eficazmente a tareas estructuradas a través de un diseño adecuado de prompts y de un proceso de fine-tuning eficiente.

Se ha elegido este modelo debido a su naturaleza generalista y su gran capacidad, lo que permite evaluar su adaptabilidad a tareas estructuradas como la generación automática de SQL. Igual que en el modelo anterior, se han utilizado dos modalidades de evaluación complementarias:

- **Inferencia directa (Zero-shot):** Se evalúa el modelo tal y como está publicado en Hugging Face, sin ningún ajuste adicional y, a diferencia del anterior, usando un prompt personalizado, diseñado exclusivamente para producir la generación de una única consulta SQL:

```
"Translate the following English question into exactly one SQL query.
Do NOT output any additional examples or explanations."

f"Question:{question}\n </s>"
```

Figura 4: Prompt personalizado para el modelo gemma-2b.

- **Fine-tuning supervisado sobre Spider:** con el objetivo de adaptar el modelo al dominio de consultas complejas, se ha aplicado un proceso de fine-tuning ligero mediante LoRA (Low-Rank Adaptation) (Hu et al., 2021).

Dada su arquitectura y tamaño, se espera que gemma-2b disponga de una mayor capacidad representacional, lo que podría permitirle generar respuestas más completas y precisas. No obstante, al tratarse de un modelo generalista, requiere ajustes adicionales y un diseño de prompts cuidadoso para lograr un rendimiento aceptable en tareas estructuradas. A través del proceso de fine-tuning, se evalúa si un modelo de estas dimensiones puede aprender estructuras SQL complejas y adaptarse eficazmente a los esquemas relacionales del dataset Spider.

3.1.3 Comparativa entre modelos

Para facilitar tanto el análisis como la interpretación de las diferencias entre los modelos seleccionados, se presenta a continuación una tabla comparativa que resume sus características más relevantes en relación con la tarea de generación automática de consultas SQL:

Tabla 2: Tabla comparativa de los modelos elegidos.

	T5-small-finetuned-wikisql	**Gemma-2b**
Arquitectura	Encoder-Decoder	Decoder-Only
Nº de parámetros	~60 millones	~2.000 millones
Preentrenamiento específico	Sí, sobre WikiSQL	No
Diseñado para text-to-SQL	Sí	No
Uso sin fine-tuning	Inferencia directa (zero-shot)	Inferencia con prompt (zero-shot)
Uso con fine-tuning (Spider)	Entrenamiento supervisado completo	Fine-tuning con LoRA

Esta comparación permite observar claramente las diferencias tanto a nivel de arquitectura interna como de propósito original. Mientas que T5-small-finetuned-wikisql representa un modelo compacto, previamente adaptado a la tarea de generación SQL sobre estructuras simples, Gemma-2b ofrece una arquitectura más potente en términos de capacidad representacional, pero sin entrenamiento previo específico para tareas estructuradas.

Este contraste permite evaluar cómo influyen la arquitectura, el tamaño del modelo y el tipo de preentrenamiento en el rendimiento final sobre la tarea text-to-SQL.

3.2 Preparación del dataset

El presente trabajo se basa en el conjunto de datos Spider (Yu et al., 2018), explicado anteriormente en el apartado 2.4.1, como base principal tanto para el proceso de fine-tuning como para la evaluación de los modelos zero-shot.

En este trabajo, se han empleado los siguientes archivos oficiales del dataset:

- Conjunto de entrenamiento (*train_spider* + *train_others*): contienen ejemplos etiquetados que cubren estructuras complejas y múltiples relaciones entre tablas. Ambos son utilizados para el proceso de fine-tuning supervisado de ambos modelos.
- Conjunto de desarrollo (*dev*): es usado como conjunto de validación, ya que no comparte esquemas con los datos de entrenamiento. Esto lo convierte en un recurso fiable para evaluar el rendimiento en condiciones realistas.

El conjunto de *test*, al no estar disponible públicamente no ha sido empleado en este trabajo.

3.3 Clasificación de complejidad de consultas

La generación de consultas text-to-SQL implica dos componentes claves: por un lado, la interpretación semántica de la instrucción natural, introducida como input del modelo, y por otro, la construcción sintáctica de la consulta en SQL, como output del modelo. Ambos aspectos pueden presentar niveles de complejidad distintos y afectar de forma diferente al rendimiento de los modelos.

Por ello, a continuación, se proponen dos clasificaciones complementarias: una basada en la complejidad del lenguaje natural de la pregunta, y otra basada en la complejidad estructural y semántica de la consulta SQL generada.

3.3.1 Lenguaje SQL

La clasificación de la complejidad del lenguaje SQL es fundamental para analizar la complejidad de las consultas de salida. Esta clasificación se basa en criterios semánticos y cognitivos, no solo de formalidad o longitud.

La complejidad de una consulta SQL depende de múltiples factores como: la presencia de condiciones (WHERE, HAVING), la agregación (SUM, AVG, etc.), la combinación de tablas (JOIN), el uso de subconsultas y la incorporación de operadores complejos (UNION, INTERSECT, EXCEPT, CASE, etc.)

Para ello, y de igual forma que en el apartado anterior, se ha definido una clasificación propia de 7 niveles, ordenados de forma ascendiente, siendo el nivel más bajo (1) la consulta simple y el más alto (7), la consulta compleja y anidada. La tabla con todos los niveles explicados de forma detallada se encuentra en el Apéndice (véase A.2).

Esta clasificación propia está diseñada de tal forma que, cuanto mayor sea la necesidad de razonar sobre la estructura de la base de datos, establecer relaciones entre múltiples entidades, o construir múltiples niveles de lógica interna, mayor es el nivel de complejidad asignado.

Para ello se considera que ciertas cláusulas como ORDER BY, DISTINCT, LIMIT, así como los modificadores ASC y DESC, no incrementan de forma sustancial la dificultad semántica de la consulta. Aunque introducen operaciones adicionales, su función se limita a aspectos de presentación o filtrado superficial del resultado, sin exigir razonamiento estructural adicional ni transformación semántica compleja.

Esta clasificación toma como ejemplo estudios como el de Shin (2022), que analiza la distancia semántica entre una instrucción en lenguaje natural y su correspondiente consulta SQL, identificando que el esfuerzo cognitivo necesario para transformaciones simples (como ordenamiento o eliminación de duplicados) es significativamente menor que el que implican relaciones entre múltiples tablas o subconsultas anidadas. Asimismo, Eyal et al. (2023) argumentan que el uso de operadores compuestos o estructuras anidadas incrementa notablemente la tasa de error, ya que requieren descomposiciones semánticas más profundas. Otros trabajos como el de Miedema et al. (2022) y el de Taipalus (2020), han demostrado que el número de tablas, condiciones lógicas complejas y estructuras jerárquicas, influyen negativamente en la formulación y comprensión de consultas SQL tanto por parte de modelos como de usuarios humanos.

3.3.2 Lenguaje natural

La clasificación de la complejidad del lenguaje natural tiene como objetivo evaluar el grado de dificultad que representa para el modelo interpretar correctamente una instrucción escrita en lenguaje humano. Esta clasificación se basa en factores lingüísticos como la longitud de la oración, el número de verbos y conectores, la presencia de cuantificadores, cláusulas condicionales o relativas, y estructuras gramaticales compuestas.

Se ha definido una clasificación propia de 7 niveles, ordenados de forma ascendiente, siendo el nivel más bajo (1) la consulta directa y corta y el más alto (7), la consulta gramaticalmente compleja. La tabla con todos los niveles explicados de forma detallada se encuentra en el Apéndice (véase A.3).

Esta clasificación permite analizar cómo la complejidad gramatical influye en la interpretación por parte del modelo. Preguntas más largas, con múltiples condiciones, pronombres relativos o estructuras subordinadas, presentan un mayor reto semántico que puede dificultar la generación correcta de la consulta SQL correspondiente. Este enfoque se inspira en trabajos como el de Gibson (1998), quien sostiene que la complejidad sintáctica, medida en función de la distancia y número de dependencias entre constituyentes, incrementa significativamente la carga cognitiva requerida para procesar una oración. Por tanto, estructuras con cláusulas relativas, coordinación o múltiples oraciones independientes no solo resultan más difíciles para humanos, sino también para modelos de lenguaje entrenados con arquitecturas secuenciales.

3.4 Fine-tuning de los modelos

En este apartado se describe el proceso de fine-tuning aplicado a los modelos seleccionados para adaptar su comportamiento al dominio de consultas SQL complejas presentes en el dataset Spider. Ambos modelos han sido entrenados con los mismos datos, tal y como se menciona en el apartado 3.2, permitiendo así una comparación directa del impacto que tiene el fine-tuning en cada arquitectura.

3.4.1 Fine-tuning de t5-small-finetuned-wikisql sobre Spider

El modelo t5-small-finetuned-wikisql ha sido previamente entrenado sobre el conjunto WikiSQL, tal y como se ha explicado en el 2.4.2, limitado a consultas simples sin estructuras complejas como subconsultas o múltiples tablas. Dado este punto de partida, se aplica un nuevo proceso de fine-tuning supervisado con el objetivo de especializar el modelo en consultas SQL estructuralmente más ricas y variadas, como las del benchmark Spider (Yu et al., 2018).

Con esta estrategia, se busca medir la capacidad de un modelo previamente ajustado a un dominio restringido para ampliar su rendimiento hacia un escenario más exigente. El análisis de los resultados obtenidos tras el fine-tuning, tal y como se muestra más adelante, permite evaluar la efectividad de esta transferencia, así como identificar posibles límites en la arquitectura encoder-decoder cuando se enfrenta a estructuras complejas.

3.4.2 Fine-tuning de Gemma-2b sobre Spider

Tal y como se detalla en el apartado 3.1.2, Gemma-2b es un modelo de tipo decoder-only, sin preajuste específico para generación de consultas SQL. Su entrenamiento generalista lo convierte en un caso de estudio adecuado para evaluar el potencial de adaptación de modelos de gran escala a tareas estructuradas mediante fine-tuning.

Para este modelo, al igual que para el anterior, se usan los conjuntos train_spider y train_others como conjuntos de entrenamiento. Con el objetivo de facilitar este proceso en un entorno con recursos limitados, se aplica la técnica LoRA, de Hu et al. (2021), que permite ajustar solo una parte reducida de los parámetros, disminuyendo el coste computacional sin comprometer significativamente el rendimiento.

El formato de entrada empleado en el ajuste, descrito previamente en el apartado 3.1.2, consiste en un prompt instructivo que dirige explícitamente al modelo a generar una única consulta SQL. Esta decisión busca maximizar la coherencia de las predicciones y facilitar su evaluación automática.

Este proceso de fine-tuning permite analizar hasta qué punto un modelo generalista, sin especialización previa, puede aprender las estructuras propias del lenguaje SQL mediante exposición directa al dominio. Su rendimiento se contrasta posteriormente con el de T5, que parte de un entrenamiento más orientado a esta tarea.

3.5 Evaluación

La evaluación de los modelos se ha llevado a cabo usando la métrica BLEU, tal y como se expone en el punto 2.5. En concreto, se ha utilizado el método 1, un enfoque de suavizado propuesto por Chen y Cherry (2014), que mejora la estabilidad de la métrica y evita penalizaciones excesivas cuando no se encuentran coincidencias en los n-gramas generados. En estos casos, si el número de n-gramas coincidentes es igual a cero, se reemplaza por un pequeño valor positivo ϵ definido empíricamente, lo que permite calcular la media geométrica sin que el resultado se anule por completo.

Antes del cálculo de la métrica BLEU, se ha aplicado un proceso de normalización sobre las consultas generadas y las de referencia, con el objetivo de eliminar diferencias superficiales que no alteran la semántica de la consulta, pero que sí podrían perjudicar la puntuación BLEU.

Este proceso incluye transformaciones sobre el formato, la puntuación, uso de espacios, comillas y símbolos. La lista completa de operaciones se detalla en el Apéndice (véase A.4).

3.6 Uso de la Inteligencia Artificial

Durante el desarrollo de este trabajo he utilizado herramientas de Inteligencia Artificial (IA) de forma complementaria para resolver dudas técnicas puntuales y agilizar tareas.

En primer lugar, me ha ayudado a estructurar el documento y a plantear qué debía incluir en cada apartado. La IA me ha permitido organizar mejor los contenidos y asegurarme de que no quedaban secciones incompletas o desordenadas.

También he utilizado IA para el diseño de las gráficas que aparecen en el capítulo de resultados. Aunque las decisiones sobre qué tipo de gráfico utilizar o qué métricas representar han sido propias, la IA me ha facilitado la personalización de elementos como leyendas, tamaños de texto, colores y etiquetas.

Otro uso puntual ha sido en el trabajo con expresiones regulares (regex), tanto en la fase de limpieza de datos como en la clasificación de consultas. En ambos casos, he identificado los patrones relevantes y he consultado a la IA la forma correcta de expresarlos en regex con Python y asegurar que se aplicaban de forma robusta.

Capítulo 4

Resultados

En este capítulo se presentan y analizan los resultados obtenidos durante este trabajo, con el objetivo de evaluar la capacidad de ambos modelos para resolver la tarea de traducción de lenguaje natural a lenguaje SQL. Las predicciones individuales generadas por cada modelo se incluyen en el Apéndice A.1.

A lo largo del capítulo, se realiza un análisis cuantitativo del rendimiento de los modelos T5-small-finetuned-wikisql y gemma-2b, tanto en su versión base como tras aplicar fine-tuning sobre el conjunto de datos Spider.

4.1 Comparativa general entre modelos

En primer lugar, se comparan los cuatro escenarios considerados: T5-small-finetuned-WikiSQL y gemma-2b, tanto su versión base como su versión con fine-tuning. El objetivo es analizar la capacidad de cada modelo para generar consultas SQL correctas, sin entrar, de momento, en detalle sobre el tipo de consulta ni la complejidad del lenguaje de entrada.

Para ello, tal y como se ha explicado previamente en el apartado 3.5, se usa la métrica BLEU, que permite cuantificar la similitud entre la consulta generada y la de referencia. Además, se calcula la desviación estándar para estimar la variabilidad de las predicciones.

A continuación, se muestra la tabla comparativa de modelos:

Tabla 3: Tabla comparativa de rendimiento de los modelos.

Modelo	Fine-tuning	BLEU (Media)	Desv. Estándar	BLEU Máx	BLEU Min	BLEU = 1
T5-small-finetuned-WikiSQL	NO	0.3669	0.2149	0.8315	0.0260	0
	SI	0.6027	0.2845	1.0000	0.0140	114
Gemma-2b	NO	0.4790	0.2406	1.0000	0.0000	26
	SI	0.5468	0.2411	1.0000	0.0097	43

A partir de estas métricas, se ha generado la Figura 5, que muestra la puntuación BLEU media por modelo y configuración, acompañada de su desviación estándar. Esta visualización permite identificar de forma clara el efecto del fine-tuning sobre ambos modelos.

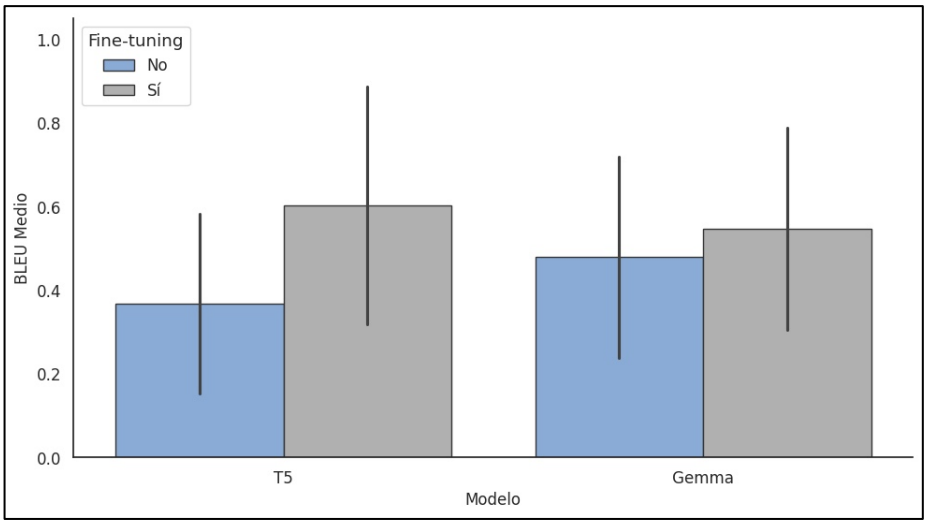


Figura 5: BLEU medio por modelo y tipo de fine-tuning.

Además, para complementar la figura anterior, se ha incluido en la Figura 6 un boxplot que representa la distribución completa de puntuaciones BLEU obtenidas por cada modelo. Esta representación evidencia la variabilidad interna en cada configuración, mostrando la mediana, los máximos, los mínimos y los cuartiles.

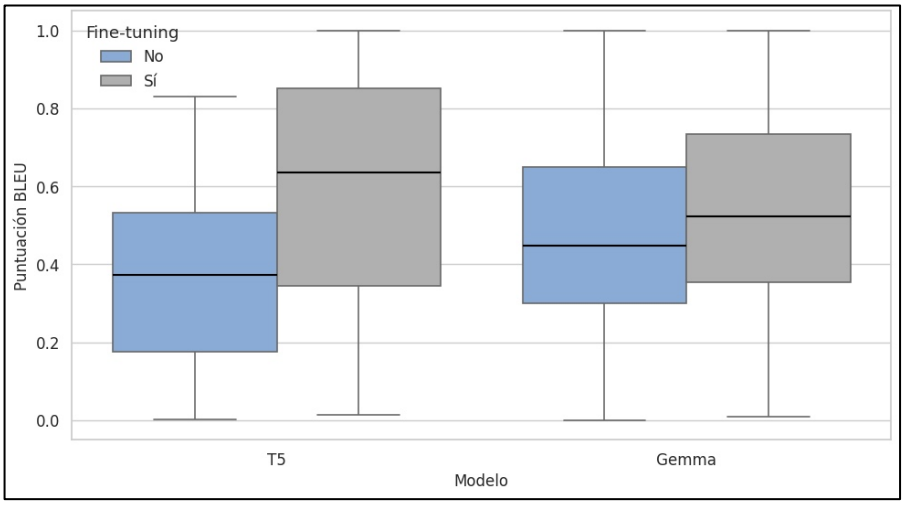


Figura 6: Distribución de puntuaciones BLEU por modelo y tipo de fine-tuning.

Por otro lado, se muestran los gráficos del entrenamiento de fine-tuning en ambos modelos. En ellos se puede ver como ha ido aprendiendo el modelo en función del Training Loss y Validation Loss.

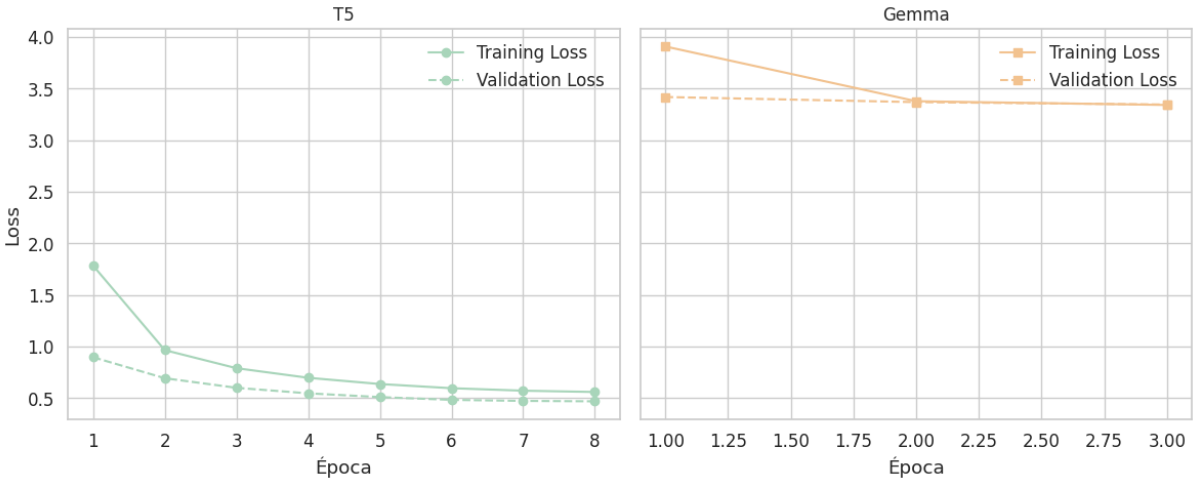


Figura 7: Evolución de la pérdida durante el fine-tuning.

Se observa que T5 muestra una disminución constante tanto en el Training Loss como en el Validation Loss, lo que indica que ambos convergen de manera progresiva sin llegar al sobreajuste. En contraste, Gemma muestra una mejora menor, llegando a la estabilidad tras la tercera época, lo cual refleja una capacidad de ajuste más limitada.

Estos resultados permiten establecer una primera visión general sobre el rendimiento de ambos modelos, evidenciando la mejora que supone el fine-tuning y sentando las bases para un análisis más detallado en función del tipo de consulta y la complejidad del lenguaje, que se desarrollará en los apartados siguientes.

4.2 Rendimiento por tipo de consulta SQL

Una vez evaluados los resultados globales, se desglosa el rendimiento de los modelos en función del tipo de consulta SQL. Con este desglose, se busca comprender mejor en qué contextos se produce un mayor o menor acierto, y si existe una relación directa entre la estructura de la consulta y la capacidad del modelo para traducir correctamente el lenguaje natural.

Para ello, se ha utilizado la clasificación propuesta en el apartado 3.3.2, que organiza las consultas en 7 niveles de forma creciente, desde instrucciones simples y directas hasta estructuras gramaticalmente más complejas.

La Figura 7 muestra la puntuación BLEU media obtenida por cada modelo, tanto en su versión base como tras aplicar fine-tuning, para cada nivel de complejidad estructural.

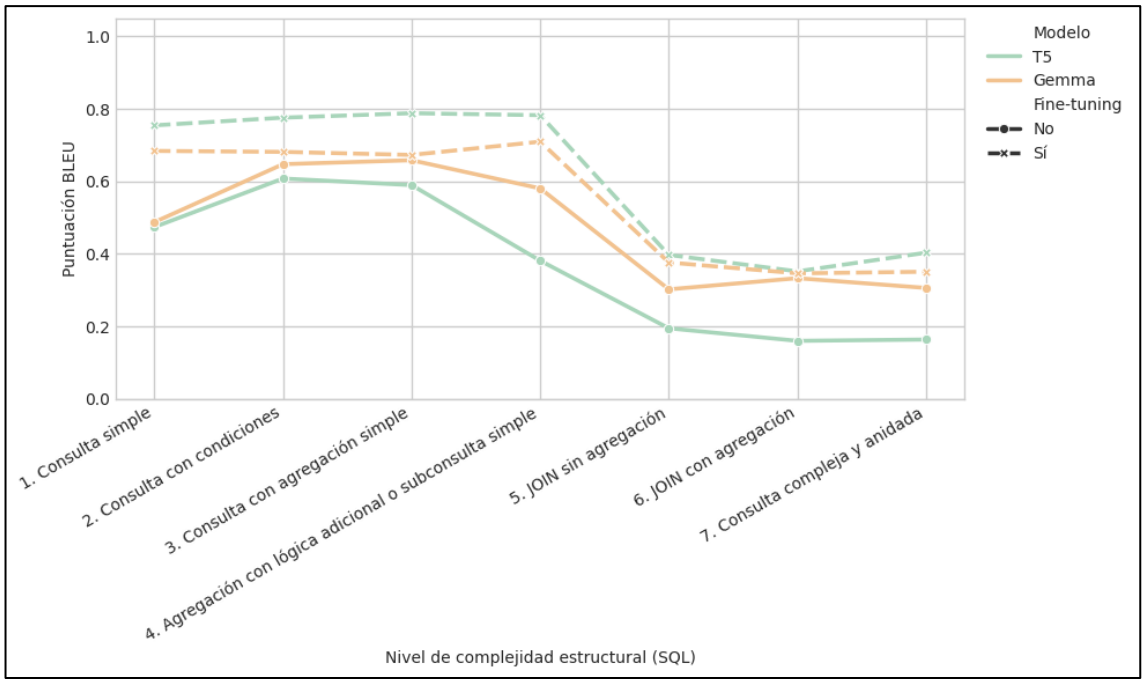


Figura 8: Puntuación BLEU media por tipo de consulta.

Los resultados muestran una tendencia descendente en el rendimiento a medida que la complejidad de las consultas aumenta.

Los primeros 4 niveles (1-4) muestran puntuaciones altas, especialmente el modelo T5 con fine-tuning, que se mantiene por encima de 0.75 en todos ellos. A partir del nivel 5, se observa una gran caída, mostrando así las dificultades de ambos niveles para tratar con varias tablas.

Para completar el análisis y ofrecer una vista más precisa, se añade a continuación una tabla donde se muestran los valores exactos de BLEU medio obtenidos por cada modelo, con y sin fine-tuning, para cada nivel de complejidad, junto con el número de muestras correspondientes.

Tabla 4: Comparativa de rendimiento BLEU por tipo de consulta SQL.

Nivel	Tipo de consulta	Nº de consultas	T5 sin fine-tuning	T5 con fine-tuning	Gemma-2b sin fine-tuning	Gemma-2b con fine-tuning
1	Consulta simple	122	0.4735	0.7541	0.4870	0.6835
2	Consulta con condiciones	130	0.6074	0.7753	0.6470	0.6809
3	Consulta con agregación simple	176	0.5893	0.7876	0.6578	0.6725
4	Agregación con lógica adicional o subconsulta simple	168	0.3806	0.7821	0.5802	0.7090
5	JOIN sin agregación	136	0.1942	0.3966	0.3016	0.3757
6	JOIN con agregación	195	0.1595	0.3512	0.3324	0.3457
7	Consulta compleja y anidada	107	0.1631	0.4026	0.3054	0.3502

Este análisis desglosado por tipo de consulta permite identificar con mayor precisión los puntos fuertes y débiles de cada modelo, y muestra cómo la complejidad estructural de la instrucción en lenguaje natural influye directamente en la calidad de la traducción a SQL.

4.3 Impacto de la complejidad lingüística en los resultados

Además de analizar la estructura de las consultas SQL, se analiza también el impacto de la complejidad lingüística de la pregunta en lenguaje natural. Para ello, se ha usado la clasificación propuesta en el apartado 3.3.1, que agrupa las preguntas en 7 niveles según su complejidad gramatical y semántica.

En la Figura 8, se muestra un gráfico del valor medio de BLEU en función del nivel de complejidad lingüística de la pregunta. Se observa claramente que, cuanto más compleja es la pregunta en lenguaje natural, más dificultades tienen los modelos para generar consultas SQL correctas. De igual forma que ocurre en la clasificación por tipo de consulta SQL, se observa que el modelo T5 con fine-tuning obtiene las puntuaciones más altas a lo largo de los diferentes niveles.

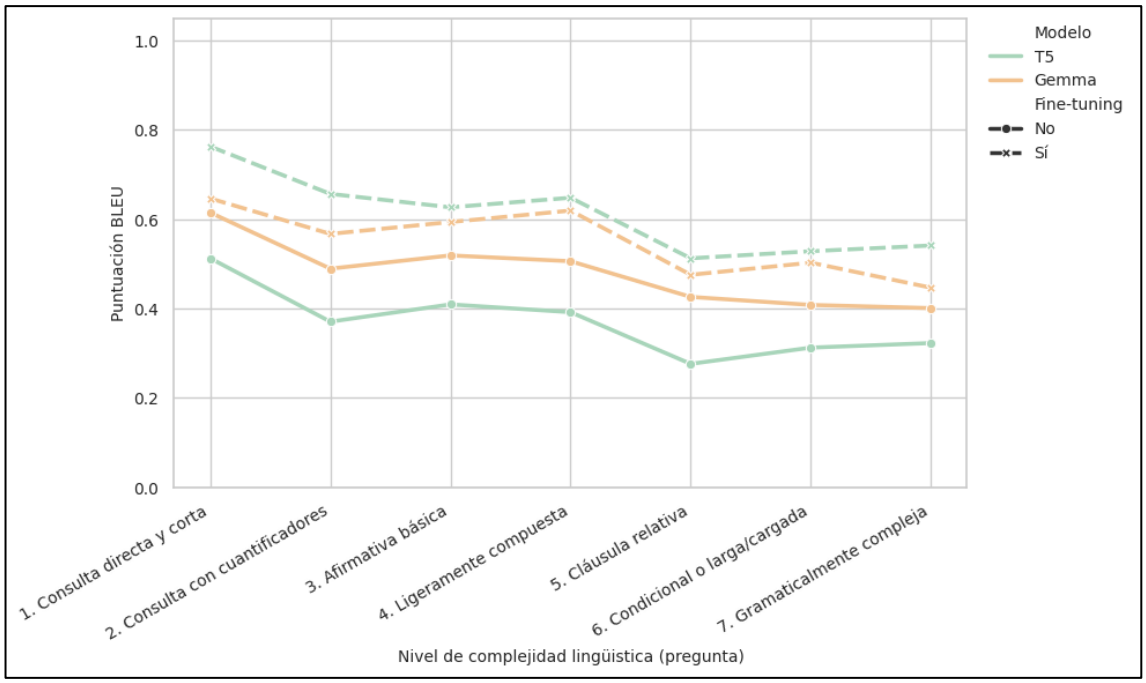


Figura 9: Rendimiento medio por nivel de complejidad lingüística.

Además, como en el apartado anterior, se incluye una tabla que muestra los valores exactos por nivel y modelo.

Tabla 5: Comparativa de rendimiento BLEU por tipo de consulta SQL.

Nivel	Tipo de pregunta	Nº de consultas	T5 sin fine-tuning	T5 con fine-tuning	Gemma-2b sin fine-tuning	Gemma-2b con fine-tuning
1	Consulta directa y corta	120	0.5125	0.7627	0.6148	0.6464
2	Consulta con cuantificadores	187	0.3706	0.6565	0.4892	0.5670
3	Afirmativa básica	189	0.4092	0.6265	0.5189	0.5936
4	Ligeramente compuesta	104	0.3918	0.6479	0.5059	0.6192
5	Cláusula relativa	183	0.2761	0.5124	0.4260	0.4755
6	Condicional o larga/cargada	108	0.3125	0.5283	0.4080	0.5026
7	Gramaticalmente compleja	143	0.3127	0.5410	0.4009	0.4467

4.4 Interacción entre complejidad lingüística y estructural

En este apartado se analiza el impacto combinado de la complejidad lingüística y la complejidad estructural sobre el rendimiento de ambos modelos. Para ello, se muestra a continuación un heatmap por cada modelo, tanto en su versión base como tras aplicarle fine-tuning, dónde se puede ver con claridad zonas de alta dificultad donde el rendimiento se ve especialmente afectado, así como evaluar la capacidad de generalización de cada modelo.

Figura 10: Heatmap de distribución de BLEU medio por complejidad (SQL vs Gramatical).

Se observa que los modelos sin fine-tuning (parte izquierda de la figura) presentan un rendimiento notablemente inferior, especialmente en combinaciones de alta complejidad (niveles 6 y 7 tanto en pregunta como en SQL). En el caso de T5 sin fine-tuning, las puntuaciones BLEU descienden hasta 0.12 en algunos puntos, mientras que con fine-tuning, alcanza el mínimo en 0.19. Por el contrario, en el modelo base se llega a un máximo de 0.65 mientras que en el modelo fine-tuned, alcanza el 0.88.

Por otro lado, el modelo Gemma sin fine-tuning muestra un rendimiento más robusto que T5, aunque sufre igualmente en las zonas de complejidad alta. Tras aplicar fine-tuning, el modelo mejora significativamente, destacando especialmente en niveles gramaticales intermedios (2–4).

Además, se observa claramente que ambos modelos, tanto antes como después de aplicar fine-tuning, se desenvuelven mejor en la primera mitad (1-4) de la clasificación por consultas SQL.

4.5 Ejemplos ilustrativos de casos exitosos y fallidos

Aunque la métrica BLEU ha sido utilizada como base para evaluar la similitud entre la consulta original y la consulta generada, en algunos casos su valor no se correlaciona con la corrección real del resultado. BLEU mide la coincidencia a nivel de tokens, pero no garantiza que la consulta sea semánticamente equivalente ni que devuelva el mismo conjunto de datos.

Para mostrar estos ejemplos, se han identificado 6 categorías de comportamiento representativo o anómalo. Para cada categoría, se ha seleccionado un ejemplo concreto, que ha sido ejecutado en la base de datos para comparar el resultado de ambas consultas. Estos ejemplos, incluyendo las consultas exactas y los outputs, se recogen en el Apéndice A.5.

Las categorías identificadas son:

Caso 1: Consultas diferentes que producen el mismo resultado

Consultas cuya sintaxis varía, pero que, al ejecutarse, devuelven exactamente el mismo resultado. A pesar de esto, BLEU penaliza esta diferencia superficial y asigna una puntuación más baja de manera injusta.

Caso 2: Consultas muy similares que devuelven resultados diferentes

Pares de consultas que solo difieren en un detalle mínimo, como un comparador o el tipo de orden (ASC, DESC), pero esa diferencia provoca que el resultado final cambie de forma significativa. Sin embargo, BLEU no comprende el impacto semántico y asigna una puntuación alta.

Caso 3: Cambios en el orden de columnas o cláusulas que no afectan el resultado.

Este caso incluye consultas donde el modelo invierte en el orden de las columnas en el SELECT, o reordena cláusulas como GROUP BY, sin alterar el contenido ni el significado de la consulta. BLEU penaliza estos cambios, aunque el resultado sea idéntico.

Caso 4: Uso innecesario de alias, DISTINCT, o cambios puramente de estilo que no alteran el output.

Consultas que introducen alias, detalles estéticos o DISTINCT de forma innecesaria. Estas transformaciones no afectan al output, pero sí disminuyen la puntuación BLEU porque se modifican los tokens generados

Caso 5: Reformulaciones semánticas profundas que BLEU no penaliza adecuadamente

En este caso se muestran consultas cuya estructura cambia por completo, como subconsultas o el uso de HAVING en lugar de WHERE, mantiene un objetivo similar. BLEU no penaliza suficientemente estos cambios, aunque puedan afectar la lógica de ejecución.

Caso 6: Diferencias menores en el nombre de tablas, como el uso de plurales, que alteran por completo el comportamiento de la consulta.

Consultas que fallan debido a pequeños errores en el nombre de la tabla, mayoritariamente el uso de plurales. Aunque el resto de la sintaxis sea correcto, esta diferencia hace que la consulta no funcione, aunque BLEU muestre un valor alto.

Estos casos refuerzan que la métrica BLEU puede resultar, en ciertos casos, engañosa tanto en sentido positivo como negativo, y muestran la necesidad de utilizar métricas complementarias para poder obtener unos resultados más precisos.

Capítulo 5

Discusión

5.1 Análisis de los resultados generales

Los resultados generales obtenidos en esta evaluación muestran diferencias claras entre ambos modelos, y gracias a ello se pueden extraer conclusiones relevantes sobre el impacto de la arquitectura, el impacto de la entrada y salida y la aplicación del fine-tuning.

En primer lugar, el modelo T5-small-finetuned-WikiSQL obtiene la mejor puntuación global tras aplicar fine-tuning, alcanzando un valor BLEU medio de 0.6027, superando al modelo Gemma-2b en las mismas condiciones (0.5468). Además, T5 consigue 114 predicciones con puntuación BLEU = 1, frente a las 43 de Gemma. Esta diferencia no solo tiene importancia estadística, sino que también se aprecia visualmente en la Figura 6, donde se observa que la mediana de BLEU tras el fine-tuning es más alta.

Una de las claves de este rendimiento está en la arquitectura del modelo. T5, al ser un modelo de tipo encoder-decoder, permite una mejor representación semántica del enunciado antes de generar la consulta SQL. Al separar tan claramente la fase de codificación del input y la de generación del output, el modelo puede interpretar mejor el objetivo de la pregunta y construir consultas más precisas. Por el contrario, Gemma-2b emplea una arquitectura decoder-only autoregresiva, que genera palabra a palabra en función del contexto anterior, lo que puede limitar su precisión en tareas que requieren estructuras formales complejas (Brown et al., 2020).

Aunque Gemma-2b tenga un tamaño mucho mayor que T5, no logra superarlo en puntuación BLEU media. Esto, probablemente se debe a que no ha sido preentrenado específicamente para tareas text-to-SQL. A esto, se le suma un detalle de gran importancia: la generación de respuestas abiertas obliga a diseñar un prompt muy preciso. En este trabajo, se ha utilizado un prompt breve, pero incluso pequeños cambios en su formulación afectan de forma considerable al resultado. Este fenómeno ha sido reportado en múltiples estudios sobre few-shot y prompt-based learning (Liu et al., 2023).

Además, al tratarse de un modelo generativo libre, ha sido necesario aplicar un proceso de extracción posterior para obtener la consulta SQL del texto generado, descartando explicaciones, comentarios o formateos incorrectos. Este paso de extracción añade un nivel de complejidad adicional y puede introducir errores si la estructura del output no es completamente predecible. En comparación, T5 al estar entrenado explícitamente para devolver solo SQL como respuesta, no requiere procesos adicionales, lo que también influye positivamente en su fiabilidad.

Por otro lado, es importante mencionar que los 8659 ejemplos del dataset Spider, pueden no haber sido suficientes para entrenar un modelo tan grande como Gemma. Aunque se aplicó fine-tuning mediante LoRA (Hu et al., 2021), esta técnica solo ajusta una pequeña parte de los parámetros del modelo, lo que puede ser insuficiente para capturar patrones complejos si no se acompaña de un mayor volumen de datos.

Aun y así, el fine-tuning ha demostrado ser positivo en ambos casos. La mejora en la puntuación BLEU media (de 0.3669 a 0.6027 en T5, y de 0.4790 a 0.5468 en Gemma) confirma la utilidad de adaptar los modelos al dominio SQL y muestra, en ambos casos, un aprendizaje de los modelos en este tipo de tarea.

En resumen, se puede decir que la arquitectura encoder-decoder es especialmente eficaz para la traducción de lenguaje natural a SQL, que el tamaño del modelo no garantiza un mejor rendimiento si no va acompañado de especialización y datos suficientes, que el fine-tuning mejora sustancialmente la capacidad del modelo, sobre todo en arquitecturas especializadas y que en modelos generativos como

Gemma, el rendimiento depende en gran parte de un prompt claro y de una extracción delicada de la consulta generada.

5.2 Efecto de la complejidad estructural y lingüística

5.2.1 Complejidad estructural: impacto en la generación SQL

En primer lugar, el análisis por tipo de consulta SQL muestra una tendencia descendente en la puntuación BLEU a medida que los niveles de dificultad aumentan. Las consultas simples y las que implican solo condiciones o agregaciones básicas obtienen los mejores resultados en ambos modelos. Por ejemplo, en el nivel 3 (agregación simple), T5 con fine-tuning alcanza un BLEU de 0.7876, mientras que en el nivel 6 (JOIN con agregación) cae hasta 0.3512.

Esta caída se debe a que las consultas de mayor nivel requieren un mayor razonamiento estructural: no basta con seleccionar una tabla o aplicar un filtro, sino que hay que identificar múltiples entidades, relaciones entre tablas y condiciones que deben combinarse correctamente.

Estas conclusiones se alinean con estudios como los de Miedema et al. (2022) y Taipalus (2020), que concluyen que el número de tablas, la presencia de condiciones lógicas complejas y la existencia de jerarquías en la estructura SQL tienen un efecto negativo tanto en humanos como en modelos de lenguaje.

5.2.2 Complejidad lingüística: impacto en la interpretación del enunciado

De forma paralela, el análisis de la complejidad lingüística muestra un patrón muy similar. A medida que la pregunta aumenta su dificultad y se introducen estructuras gramaticales más complejas como cláusulas relativas, condicionales o formulaciones gramaticalmente cargadas, el rendimiento de los modelos disminuye.

Por ejemplo, las preguntas de nivel 1 generan resultados altos (BLEU de 0.7627 en T5 con fine-tuning), mientras que aquellas con estructuras complejas (nivel 7) descienden hasta 0.5410. En el caso de Gemma, la caída es aún más clara, lo que indica que los modelos no solo se ven afectados por la estructura de la salida, sino también por las dificultades de comprensión semántica del input.

Este efecto se apoya en trabajos como los de Gibson (1998), quien propone que la carga cognitiva aumenta proporcionalmente con el número de dependencias sintácticas y la distancia entre constituyentes en una oración.

5.2.3 Interacción entre complejidad lingüística y estructural

En el heatmap de la Figura 10, se observa claramente el efecto combinado entre ambas complejidades, donde las zonas de menor rendimiento BLEU están concentradas en la parte inferior derecha, cuando tanto la pregunta como la consulta se sitúan en niveles de alta complejidad.

Por ejemplo, T5 sin fine-tuning obtiene valores cercanos a 0.12-0.19 en combinaciones de nivel 6-7, mientras que T5 con fine-tuning mejora hasta aproximadamente 0.41, pero sigue lejos de los máximos obtenidos en combinaciones de baja complejidad.

Esto sugiere que la dificultad no es simplemente acumulativa, sino que la interacción entre ambas dimensiones agrava el problema, tal y como indican estudios recientes como los de Eyal et al. (2023) y Shin (2022). Ambos autores coinciden en que los errores tienden a multiplicarse cuando un modelo debe razonar simultáneamente sobre semántica natural y lógica estructurada.

En definitiva, estos resultados refuerzan la necesidad de adaptar los modelos no solo al dominio estructurado mediante fine-tuning, sino también de prestar atención a la complejidad del lenguaje natural de la entrada.

5.3 Limitaciones

A pesar de los resultados obtenidos y de las mejoras evidentes tras aplicar fine-tuning, este trabajo tiene varias limitaciones que necesitan una mención:

En primer lugar, tal y como se ha mencionado anteriormente, el volumen de datos utilizado para el entrenamiento ha sido limitado en relación con el tamaño y la capacidad de los modelos evaluados, especialmente en el caso de Gemma-2b.

Además, es necesario tener en cuenta que el conjunto de evaluación empleado ha sido únicamente el conjunto de validación (dev) del dataset Spider, sin acceso al conjunto de test oficial. Esto implica que, si bien los resultados permiten comparar entre modelos bajo condiciones similares, no pueden considerarse como una medición absoluta del rendimiento frente a otros trabajos del estado del arte que sí utilizan dicho conjunto.

Por otro lado, la métrica utilizada para la evaluación ha sido BLEU, que, aunque sea útil para medir la similitud entre la consulta generada y la de referencia, no garantiza que la predicción sea funcionalmente correcta. BLEU no evalúa si la consulta devuelve el resultado esperado al ejecutarse, ni si es sintácticamente válida en todos los casos. Por ello, otras métricas como Exact Match o Execution Accuracy (Yu et al., 2018) podrían haber aportado una visión más completa del rendimiento real.

Capítulo 6

Conclusiones y trabajo futuro

6.1 Conclusiones

Este trabajo ha permitido explorar y comparar el rendimiento de dos modelos de lenguaje de arquitecturas diferentes en la tarea de traducción de lenguaje natural a SQL, o lo que es lo mismo, text-to-SQL.

El modelo T5-small-finetuned-WikiSQL, basado en una arquitectura encoder-decoder y entrenado previamente sobre WikiSQL, ha mostrado mejores resultados, especialmente tras aplicar fine-tuning sobre el dataset Spider. Más concretamente, ha alcanzado una puntuación BLEU media de 0.6027 y ha generado 114 predicciones idénticas a la referencia, superando al modelo Gemma-2b, que ha obtenido un BLEU medio de 0.5468 y 43 predicciones exactas.

Por un lado, la arquitectura encoder-decoder parece la más adecuada para tareas estructuradas, ya que permite separar claramente el procesamiento del input y la generación del output, facilitando una representación semántica más precisa. Por otro lado, se puede confirmar que el fine-tuning sobre un dominio específico (en este caso, SQL complejo) mejora significativamente el rendimiento.

A pesar de la gran diferencia de tamaño de Gemma-2b, su rendimiento no ha superado al de T5-small, lo cual demuestra que únicamente el tamaño del modelo no garantiza mejores resultados. La falta de especialización previa y la necesidad de un prompt bien diseñado, han sido algunos de los límites que ha tenido este modelo.

Uno de los factores clave de este trabajo ha sido el desarrollo de una clasificación de complejidad doble, una centrada en el lenguaje natural y otra en la estructura de la consulta SQL. Esta clasificación ha permitido analizar con mayor detalle cómo influyen distintos niveles de dificultad en el rendimiento de los modelos. Los resultados muestran claramente que, tanto a nivel gramatical como estructural, el rendimiento cae a medida que aumenta la complejidad, y que las combinaciones más difíciles son las que presentan menor tasa de acierto.

En definitiva, este trabajo ha sido de utilidad para entender mejor el papel que juegan la arquitectura del modelo, el tipo de entrenamiento previo y la complejidad del lenguaje en la traducción automática de lenguaje natural a SQL.

6.2 Próximos pasos

Aunque los resultados finales de este trabajo han sido satisfactorios y coherentes, hay varias propuestas de trabajos o investigaciones que se podrían realizar en el futuro:

- **Ampliar el dataset de entrenamiento**, tener un mayor número de muestras para poder entrenar los modelos ayudará a entender si el factor limitante de Gemma-2b son los datos.
- **Aplicar métricas de evaluación más completas**, como Exact Match o Execution Accuracy, que evalúan la validez funcional de la consulta generada, no solo su similitud superficial.
- **Explorar nuevos formatos de prompt**, especialmente en modelos decoder-only como Gemma, dónde la generación depende en gran medida de los prompts de entrada.
- **Diseñar una interfaz de conversación**, donde los usuarios sin conocimientos o con conocimientos básicos de SQL puedan interactuar con bases de datos mediante preguntas en

lenguaje natural, integrando alguno de estos modelos como backend y poder evaluar su utilidad en entornos reales.

En conclusión, este trabajo abre la puerta a múltiples líneas de investigación que podrían enriquecer y profundizar en la relación entre lenguaje natural y bases de datos.

Bibliografia

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions On Neural Networks*, *5*(2), 157-166. https://doi.org/10.1109/72.279181

Bhaskar, A., Tomar, T., Sathe, A., & Sarawagi, S. (2023). Benchmarking and Improving Text-to-SQL Generation under Ambiguity. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2310.13659

Brown, P. F., deSouza, P. V., Mercer, R. L., Della Pietra, V. J., & Lai, J. C. (1992). Class-based n -gram models of natural language. *Computational Linguistics*, *18*(4), 467-479. https://doi.org/10.5555/176313.176316

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., . . . Amodei, D. (2020). Language Models are Few-Shot Learners. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2005.14165

Cao, R., Chen, L., Chen, Z., Zhao, Y., Zhu, S., & Yu, K. (2021). LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2106.01093

Chen, B., & Cherry, C. (2014). A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU. *Proceedings Of The Ninth Workshop On Statistical Machine Translation*. https://doi.org/10.3115/v1/w14-3346

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1103.0398

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1810.04805

Elgohary, A., Hosseini, S., & Awadallah, A. H. (2020). Speak to your Parser: Interactive Text-to-SQL with Natural Language Feedback. *Proceedings Of The 58th Annual Meeting Of The Association For Computational Linguistics*. https://doi.org/10.18653/v1/2020.acl-main.187

Eyal, B., Bachar, A., Haroche, O., Mahabi, M., & Elhadad, M. (2023). Semantic Decomposition of Question and SQL for Text-to-SQL Parsing. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2310.13575

Finegan-Dollak, C., Kummerfeld, J. K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., & Radev, D. (2018). Improving Text-to-SQL evaluation methodology. *Proceedings Of The 60th Annual Meeting Of The Association For Computational Linguistics (Volume 1: Long Papers)*, 351-360. https://doi.org/10.18653/v1/p18-1033

Gan, Y., Chen, X., & Purver, M. (2021). Exploring Underexplored Limitations of Cross-Domain Text-to-SQL Generalization. *Proceedings Of The 2021 Conference On Empirical Methods In Natural Language Processing*. https://doi.org/10.18653/v1/2021.emnlp-main.702

Gibson, E. (1998). Linguistic complexity: locality of syntactic dependencies. *Cognition*, *68*(1), 1-76. https://doi.org/10.1016/s0010-0277(98)00034-1

Giordani, A., & Moschitti, A. (2012). Automatic Generation and Reranking of SQL-Derived Answers to NL Questions. En *Communications in computer and information science* (pp. 59-76). https://doi.org/10.1007/978-3-642-45260-4_5

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J., Liu, T., & Zhang, D. (2019). Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1905.08205

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term memory. *Neural Computation*, *9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2106.09685

Hugging Face – The AI community building the future. (s. f.). https://huggingface.co/

Iyer, S., Konstas, I., Cheung, A., Krishnamurthy, J., & Zettlemoyer, L. (2017). Learning a Neural Semantic Parser from User Feedback. *Proceedings Of The 60th Annual Meeting Of The Association For Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/p17-1089

Kim, H., So, B. H., Han, W. S., & Lee, H. (2020). Natural language to SQL: Where are we today?. Proceedings of the VLDB Endowment, 13(10), 1737-1750.

Martin, J. H., & Jurafsky, D. (2009). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (Vol. 23). Upper Saddle River: Pearson/Prentice Hall.

Li, F., & Jagadish, H. V. (2014). Constructing an interactive natural language interface for relational databases. *Proceedings Of The VLDB Endowment*, *8*(1), 73-84. https://doi.org/10.14778/2735461.2735468

Meister, C., & Cotterell, R. (2021). Language model evaluation beyond perplexity. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2106.00085

Miedema, D., Fletcher, G., & Aivaloglou, E. (2022). So many brackets! *ICPC '22: Proceedings Of The 30th IEEE/ACM International Conference On Program Comprehension*, 122-132. https://doi.org/10.1145/3524610.3529158

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Interspeech 2022*. https://doi.org/10.21437/interspeech.2010-343

Nascimento, E., García, G., Feijó, L., Victorio, W., Izquierdo, Y., De Oliveira, A. R., Coelho, G., Lemos, M., Garcia, R., Leme, L., & Casanova, M. (2024). Text-to-SQL Meets the Real-World. *In Proceedings Of The 26th International Conference On Enterprise Information Systems (ICEIS 2024) - Volume 1, Pages 61-72*. https://doi.org/10.5220/0012555200003690

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., & Mian, A. (2023). A Comprehensive Overview of Large Language Models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2307.06435

OpenAI. (2023). GPT-4 Technical Report. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2303.08774

Papineni, K., Roukos, S., Ward, T., & Zhu, W. (2001). BLEU. *Proceedings Of The 40th Annual Meeting Of The Association For Computational Linguistics*, 311. https://doi.org/10.3115/1073083.1073135

Popescu, A., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. *IUI '03: Proceedings Of The 8th International Conference On Intelligent User Interfaces*. https://doi.org/10.1145/604045.604070

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1910.10683

Rajkumar, N., Li, R., & Bahdanau, D. (2022). Evaluating the Text-to-SQL Capabilities of Large Language Models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2204.00498

Reiter, E. (2018). A Structured Review of the Validity of BLEU. *Computational Linguistics*, *44*(3), 393-401. https://doi.org/10.1162/coli_a_00322

Ren, S., Guo, D., Lu, S., Zhou, L., Liu, S., Tang, D., Sundaresan, N., Zhou, M., Blanco, A., & Ma, S. (2020). CodeBLEU: a Method for Automatic Evaluation of Code Synthesis. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2009.10297

Scholak, T., Schucher, N., & Bahdanau, D. (2021). PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2109.05093

Shin, S. (2022). Effect of semantic distance on learning structured query language: An empirical study. *Frontiers In Psychology*, *13*. https://doi.org/10.3389/fpsyg.2022.996363

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1409.3215

Taipalus, T. (2020). The effects of database complexity on SQL query formulation. *Journal Of Systems And Software*, *165*, 110576. https://doi.org/10.1016/j.jss.2020.110576

Tang, L. R., & Mooney, R. J. (2000). Automated construction of database interfaces. *2000 Joint SIGDAT Conference On Empirical Methods In Natural Language Processing And Very Large Corpora*, *13*, 133-141. https://doi.org/10.3115/1117794.1117811

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., Tafti, P., Hussenot, L., Chowdhery, A., Roberts, A., Barua, A., Botev, A., Castro-Ros, A., Slone, A., Héliou, A., . . . Kenealy, K. (2024). Gemma: Open Models Based on Gemini Research and Technology. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2403.08295

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1706.03762

Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2019). RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1911.04942

Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017). SQLizer: query synthesis from natural language. *Proceedings Of The ACM On Programming Languages*, *1*(OOPSLA), 1-26. https://doi.org/10.1145/3133887

Yin, P., & Neubig, G. (2017). A Syntactic Neural Model for General-Purpose Code Generation. *Proceedings Of The 60th Annual Meeting Of The Association For Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/p17-1041

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., Ma, J., Li, I., Yao, Q., Roman, S., Zhang, Z., & Radev, D. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1809.08887

Zelle, J. M., & Mooney, R. J. (1996, August). Learning to parse database queries using inductive logic programming. In Proceedings of the national conference on artificial intelligence (pp. 1050-1055).

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1709.00103

Zhu, X., Li, Q., Cui, L., & Liu, Y. (2024). Large Language Model Enhanced Text-to-SQL Generation: A survey. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2410.06011

Apéndice

A.1 Código fuente y resultados completos

El código completo y los resultados de este trabajo están disponibles en GitHub:
https://github.com/MBacaicoa44/Del_NL_a_SQL

A.2 Clasificación de lenguaje SQL

Tabla A-2: Tabla de clasificación por tipo de consulta SQL.

Nivel	Etiqueta	Descripción	Ejemplo
1	Consulta simple	Consulta simple, sin condiciones ni estructuras complejas. Se incluyen cláusulas como ORDER BY, LIMIT, DISTINCT, ASC, DESC, al no aportar complejidad semántica significativa.	SELECT name FROM employee ORDER BY age
2	Consulta con condiciones	Consulta con condiciones simples, usando WHERE sin subconsultas ni lógica adicional.	SELECT name FROM Highschooler WHERE grade = 10
3	Consulta con agregación simple	Consulta con funciones de agregación básicas sin filtros ni agrupaciones.	SELECT count(*) FROM Courses
4	Agregación con lógica adicional o subconsulta simple	Agregación con lógica adicional: uso de GROUP BY, HAVING, o subconsultas simples.	SELECT template_id , count (*) FROM Documents GROUP BY template_id
5	JOIN sin agregación	Combinación de tablas mediante JOIN, sin agregación.	SELECT T1.Airline FROM AIRLINES AS T1 JOIN FLIGHTS AS T2 ON T1.uid = T2.Airline WHERE T2.DestAirport = “AHD"
6	JOIN con agregación	JOIN con agregación: requiere estructurar relaciones y aplicar funciones de resumen.	SELECT T1.ContId , T1.Continent , count(*) FROM CONTINENTS AS T1 JOIN COUNTRIES AS T2 ON T1.ContId = T2.Continent GROUP BY T1.ContId;

7	Consulta compleja y anidada	Consulta compleja y anidada: múltiples subconsultas, JOIN + subconsulta, operadores UNION, INTERSECT, CASE, etc.	SELECT DISTINCT T2.semester_id FROM Degree_Programs AS T1 JOIN Student_Enrolment AS T2 ON T1.degree_program_id = T2.degree_program_id WHERE degree_summary_name = 'Master' INTERSECT SELECT DISTINCT T2.semester_id FROM Degree_Programs AS T1 JOIN Student_Enrolment AS T2 ON T1.degree_program_id = T2.degree_program_id WHERE degree_summary_name = 'Bachelor'

A.3 Clasificación de lenguaje natural

Tabla A-3: Tabla de clasificación por pregunta en lenguaje natural.

Nivel	Etiqueta	Descripción	Ejemplo
1	Consulta directa y corta	≤ 7 palabras	What is the total number of singers?
2	Consulta con cuantificadores	Contiene palabras como all, any, every, none	Show name, country, age for all singers ordered by age from the oldest to the youngest.
3	Afirmativa básica	Enunciado general sin elementos destacados	Find the number of distinct type of pets.
4	Ligeramente compuesta	Presencia de and, or, between, having, where, join, group, order	Find the type and weight of the youngest pet.
5	Cláusula relativa	Presencia de pronombres relativos (who, which, that) o cláusulas relativas	Find the first name of students who have cat or dog pet.
6	Condicional o larga/cargada	Contiene if o más de 15 palabras	Return the different document ids along with the number of paragraphs corresponding to each, ordered by id.
7	Gramaticalmente compleja	≥ 2 oraciones o ≥ 2 entidades reconocidas	For students who have pets , how many pets does each student have ? list their ids instead of names .

A.4 Criterios de normalización

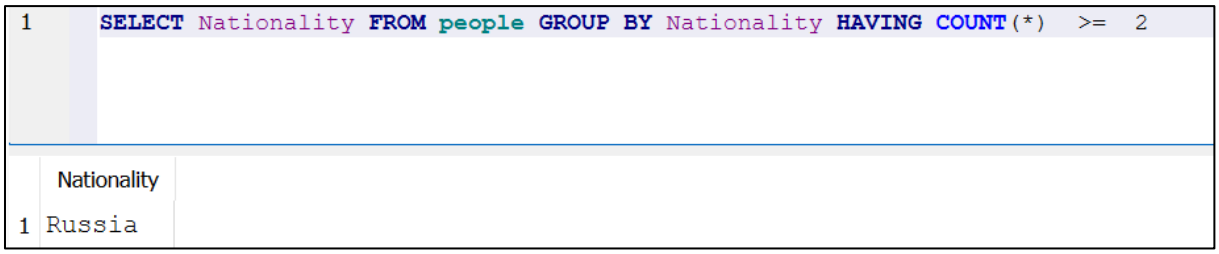
Tabla A.4: Tabla de elementos normalizados.

Tipo de normalización	Descripción
Conversión a minúsculas	Homogeneiza mayúsculas/minúsculas para evitar diferencias artificiales.
Eliminación del punto y coma final	Elimina **;** al final de la consulta si no aporta información estructural.
Unificación de comillas	Reemplaza comillas dobles y tipográficas por comillas simples '
Eliminación de *backticks*	Borra los símbolos ` utilizados en algunos motores SQL.
Reducción de espacios múltiples	Sustituye varios espacios por uno solo.
Eliminación de espacios alrededor de símbolos	Quita espacios antes y después de **,** , **=, (**, **)**, etc.
Inserción de espacios tras operadores	Añade espacio tras =, **,** , **>** si van pegados a palabras o números.
Inserción de espacios entre tokens compuestos	Mejora la segmentación entre operadores y elementos sintácticos.
Eliminación de paréntesis redundantes	Suprime expresiones como (a = b) si no aportan complejidad estructural.
Corrección de espacios en valores entre comillas	Ejemplo: ' ca ' → 'ca'.
Inserción condicional de espacios en funciones	Mantiene avg() intacto, pero transforma name(en name (si no es función.
Limpieza final de espacios	Elimina espacios innecesarios al inicio o final de la consulta.

A.5 Ejemplos ilustrativos reales

A.5.1 Caso 1: Consultas diferentes que producen el mismo resultado

Puntuación BLEU: 0.9611



Consulta SQL original

```
1  SELECT nationality FROM people GROUP BY nationality HAVING count(*) > 1
```

	Nationality
1	Russia

Consulta SQL generada

A.5.2 Caso 2: Consultas muy similares que devuelven resultados diferentes

Puntuación BLEU: 0.9602

```
1  LANGUAGE ,  count(*) FROM TV_Channel GROUP BY LANGUAGE ORDER BY count(*) ASC LIMIT 1;
```

	Language	count(*)
1	English	3

Consulta SQL original

```
1  SELECT language, count(*) FROM tv_channel GROUP BY language ORDER BY count(*) DESC LIMIT 1;
```

	Language	count(*)
1	Italian	12

Consulta SQL generada

A.5.3 Caso 3: Cambios en el orden de columnas o cláusulas que no afectan el resultado

Puntuación BLEU: 0.9133

```
SELECT count(*) , LOCATION FROM shop GROUP BY LOCATION
```

	count(*)	Location
1	1	Espoo
2	1	Helsinki
3	1	Jakobstad
4	1	Kotka
5	1	Kuopio
6	1	Lahti
7	1	Mariehamn
8	1	Turku
9	1	Valkeakoski

Consulta SQL original

```
SELECT location, count(*) FROM shop GROUP BY location;
```

	Location	count(*)
1	Espoo	1
2	Helsinki	1
3	Jakobstad	1
4	Kotka	1
5	Kuopio	1
6	Lahti	1
7	Mariehamn	1
8	Turku	1
9	Valkeakoski	1

Consulta SQL generada

A.5.4 Caso 4: Uso innecesario de alias, DISTINCT, o cambios puramente de estilo que no alteran el output

Puntuación BLEU: 0.7048

```
SELECT Age , Hometown FROM teacher
```

	Age	Hometown
1	32	Blackrod Urban District
2	29	Bolton County Borough
3	26	Farnworth Municipal Borough
4	33	Horwich Urban District
5	45	Kearsley Urban District
6	41	Little Lever Urban District
7	39	Turton Urban District

Consulta SQL original

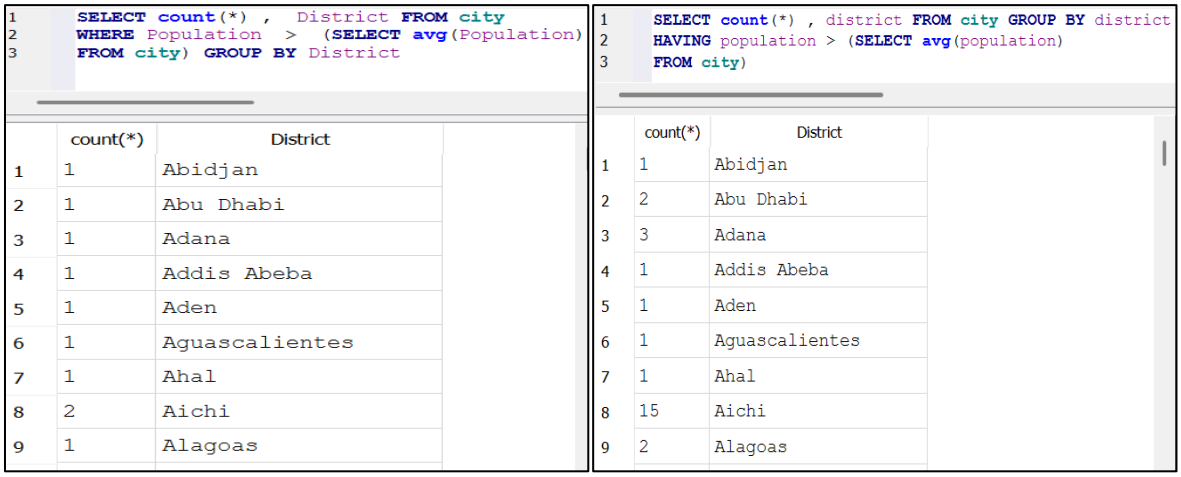
```
SELECT t.age, t.hometown FROM teacher AS t;
```

	Age	Hometown
1	32	Blackrod Urban District
2	29	Bolton County Borough
3	26	Farnworth Municipal Borough
4	33	Horwich Urban District
5	45	Kearsley Urban District
6	41	Little Lever Urban District
7	39	Turton Urban District

Consulta SQL generada

A.5.5 Caso 5: Reformulaciones semánticas profundas que BLEU no penaliza adecuadamente

Puntuación BLEU: 0.9172



```
SELECT count(*) , District FROM city
WHERE Population > (SELECT avg(Population)
FROM city) GROUP BY District
```

	count(*)	District
1	1	Abidjan
2	1	Abu Dhabi
3	1	Adana
4	1	Addis Abeba
5	1	Aden
6	1	Aguascalientes
7	1	Ahal
8	2	Aichi
9	1	Alagoas

```
SELECT count(*) , district FROM city GROUP BY district
HAVING population > (SELECT avg(population)
FROM city)
```

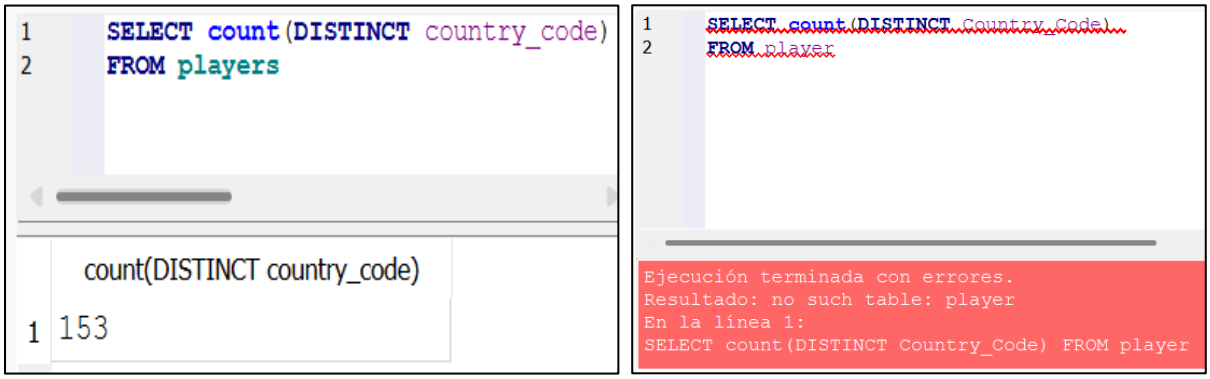
	count(*)	District
1	1	Abidjan
2	2	Abu Dhabi
3	3	Adana
4	1	Addis Abeba
5	1	Aden
6	1	Aguascalientes
7	1	Ahal
8	15	Aichi
9	2	Alagoas

Consulta SQL original

Consulta SQL generada

A.5.6 Caso 6: Diferencias menores en el nombre de tablas, como el uso de plurales, que alteran por completo el comportamiento de la consulta

Puntuación BLEU: 0.9789



```
SELECT count(DISTINCT country_code)
FROM players
```

	count(DISTINCT country_code)
1	153

```
SELECT count(DISTINCT Country_Code)
FROM player
```

```
Ejecución terminada con errores.
Resultado: no such table: player
En la línea 1:
SELECT count(DISTINCT Country_Code) FROM player
```

Consulta SQL original

Consulta SQL generada