



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

**“Modelo generativo de SQL a partir de
consultas en español”**

2024 – A104

Presentan

Victor Ulises Miranda Chávez

Adair Nicolas Hernández

Ives Lancelote Pérez Sánchez

Zury Yael Rubio López

Directores

**M. en C. Enrique Alfonso Carmona
García**

**M. en C. Ituriel Enrique Flores
Estrada**

INSTITUTO POLITÉCNICO NACIONAL



Junio 2024



**INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO
SUBDIRECCIÓN ACADÉMICA**



No. de TT: 2024 - A104

10/Junio/2024

Documento Técnico

**“Modelo generativo de SQL a partir de
consultas en español”**

Presentan

Victor Ulises Miranda Chávez ¹

Adair Nicolas Hernández ²

Ives Lancelote Pérez Sánchez ³

Zury Yael Rubio López ⁴

Directores

**M. en C. Enrique Alfonso Carmona
García**

**M. en C. Ituriel Enrique Flores
Estrada**

Resumen

En este trabajo, se presenta una propuesta basada en reglas para abordar la traducción automática de consultas de lenguaje natural a consultas SQL. Basándonos en técnicas de procesamiento del lenguaje natural, nuestro modelo captura la estructura de las consultas en lenguaje natural, y las convierte en una serie de posibles consultas SQL equivalentes. El rendimiento obtenido por nuestro modelo utilizando un conjunto de datos basado en el conjunto “Spider”, el cual fue traducido al español y limitado a consultas de nivel 1, fue del 25% bajo la métrica Top-N accuracy, lo que revela áreas de mejora, las cuales se describen en este trabajo.

Palabras clave: Bases de Datos, Inteligencia Artificial, Procesamiento del Lenguaje Natural, SQL

¹ ulimirandachavez@hotmail.com

² adairnicolas1@gmail.com

³ lancelote.ps@gmail.com

⁴ ryzust@gmail.com

CARTA RESPONSIVA

Que otorga visto bueno y avala la conclusión de documentación del Trabajo Terminal bajo los lineamientos establecidos por la Comisión Académica de Trabajos Terminales (CATT)

CDMX, a 17 de Junio de 2024

M. EN C. ANDRÉS ORTIGOZA CAMPOS
PRESIDENTE DE LA COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES
P R E S E N T E

EN ATENCIÓN A:
M. EN A. N. MARÍA MAGDALENA SALDÍVAR ALMOREJO
SECRETARIA EJECUTIVA

Por medio de la presente, se informa que el Trabajo Terminal Núm. 2024-A104

Que lleva por Título: Modelo generativo de SQL a partir de consultas en
español

Fue concluido satisfactoriamente por:

Miranda Chávez Victor Ulises
Nicolas Hernández Adair
Pérez Sánchez Ives Lancelote
Rubio López Zury Yael

Se avala que la documentación entregada mediante discos en formato DVD fue **revisada de manera precisa y exhaustiva** con el propósito de asegurar que los avances desarrollados bajo la supervisión de quien o quienes suscriben, hayan cumplido con lo planteado en el protocolo original, así como en lo establecido por el Documento Rector de Operación y Evaluación para los Trabajos Terminales de la ESCOM.

ATENTAMENTE
"LA TÉCNICA AL SERVICIO DE LA PATRIA"



M. en C. Carmona García Enrique Alfonso
Director/a



Ituriel C. Flores Estrada
M. en C. Flores Estrada Ituriel Enrique
Director/a

SUBDIRECCIÓN ACADÉMICA
DEPARTAMENTO DE FORMACIÓN INTEGRAL E INSTITUCIONAL
CATT COMISIÓN
ACADÉMICA DE
TRABAJOS TERMINALES

FORMATO:

CARTA RESPONSIVA

ADVERTENCIA

“Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan.”

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

ÍNDICE

INTRODUCCIÓN.....	8
CAPÍTULO 1. Preámbulo.....	9
1.1 Situación problemática	9
1.2 Justificación.....	10
1.3 Objetivos	11
1.3.1 Objetivo general.....	11
1.3.2 Objetivos específicos	11
1.4 Metodología	12
1.4.1 Metodología CRISP-ML	12
CAPÍTULO 2. Estado del arte	14
2.1 Revisión de investigaciones anteriores en generación automática de SQL	15
CAPÍTULO 3. Marco teórico	20
3.1 Lenguaje SQL	20
3.1.1 Estructura básica de una consulta SQL	20
3.1.2 Componentes de una consulta SQL.....	21
3.2 Aprendizaje de máquina.....	22
3.2.1 Introducción al aprendizaje de máquina.....	22
3.2.2 Reconocimiento de patrones.....	23
3.2.3 Inducción de reglas.....	24
3.2.4 Algoritmo Quasi-Optimal (AQ).....	24
3.3 Procesamiento de Lenguaje Natural (PLN)	29
3.3.1 Introducción al procesamiento de lenguaje natural	29
3.3.2 Tratamiento de palabras en el PLN.....	29
3.3.2 Tuberías de procesamiento en PLN.....	30
3.3.2.1 Diferencia de Tubería y Preprocesamiento.....	30
3.3.3 Técnicas de generación de incrustaciones.....	31
3.3.4 Desafíos en la generación de incrustaciones coherentes y relevantes.....	32
3.4 Introducción a la tarea text-to-SQL	32
3.4.1 Text to SQL.....	32
3.4.2 Variantes de la tarea text to SQL	33
3.4.3 Conjuntos de datos	33
3.4.5 Etiquetado y anotación de datos	34

3.4.6 Métricas de evaluación, como Execution Accuracy y Logical Form Accuracy	34
3.5 Desafíos y Limitaciones.....	36
3.5.1 Ambigüedad en consultas en lenguaje natural	36
CAPÍTULO 4. Análisis y diseño del modelo	37
4.1 Alcance	37
4.2 Requerimientos funcionales del modelo	38
4.3 Arquitectura del modelo.....	38
4.3.1 Identificador de secciones relevantes	39
4.3.2 Tratamiento de secciones extraídas.....	39
4.3.3 Mapeo a plantilla SQL.....	40
4.3.4 Evaluador semántico	41
4.4 Restricciones del modelo	41
4.5 Tecnologías a utilizar.....	42
4.6 Análisis de costos.....	42
4.7 Análisis de riesgos	43
CAPÍTULO 5. Implementación del modelo.....	45
5.1 Recopilación de datos	45
5.1.1 Evaluación y selección de datasets.....	45
5.1.2 Recopilar el dataset	46
5.1.3 Análisis profundo de Spider	46
5.1.4 Preprocesamiento del Spider para su posterior tratamiento de traducción.....	47
5.1.5 Traducción automática.....	47
5.1.5.1 Corrección de traducciones incorrectas	48
5.2 Anotación de las consultas en lenguaje natural	48
5.2.1 Etiquetas de sección	49
5.2.2 Etiquetas por valor	49
5.3 Desarrollo del Modelo.....	50
5.3.1 Selección de algoritmos de aprendizaje automático.....	50
5.3.1.2 Resultados de nuestra implementación del algoritmo Quasi-Optimal (AQ) Learning para la obtención de reglas	50
5.3.2 Diseño y creación de tubería de datos	51
5.3.2.1 TextPipeline.....	51
5.3.2.2 SimplePipeline.....	52
5.3.2.3 EmbeddingPipeline.....	52

5.3.3 Manipulación post-entrenamiento de los resultados obtenidos por nuestra implementación del AQ.....	53
5.3.1 Extracción de palabras delimitadoras de secciones (reglas).....	53
5.3.2 Generación de la consulta	53
5.3.4 Descripción general de la transformación de una consulta en lenguaje natural a una sentencia SQL con un ejemplo:	55
CAPÍTULO 6. Pruebas y validación	58
6.1 Preparación de los datos de prueba	58
6.1.1 Proceso de preparación con los datos recuperados SPIDER.....	58
6.1.1 Proceso de preparación con datos controlados	59
6.2 Validar el rendimiento del modelo.....	60
6.2.1 Evaluación Top-N Accuracy	60
6.3 Resultados	60
6.3.1 Resultados obtenidos.....	60
6.3.2 Análisis	62
CONCLUSIONES	62
TRABAJO A FUTURO	62
REFERENCIAS	64
GLOSARIO	67
APÉNDICES	69
Apéndice 1. Índice de figuras	69
Apéndice 2. Índice de tablas	70
Apéndice 3: Posible aplicación práctica del modelo como parte de un sistema.....	71

INTRODUCCIÓN

Este proyecto terminal se centra en el desarrollo de un modelo de Inteligencia artificial que aborde los desafíos asociados con la traducción automática de consultas en lenguaje natural a consultas estructuradas en lenguaje SQL. Esta iniciativa nace en respuesta a la necesidad de mejorar la accesibilidad y usabilidad de las bases de datos mediante interfaces más intuitivas y eficientes.

Este trabajo de investigación se fundamenta en la importancia de facilitar la interacción entre usuarios y sistemas de gestión de bases de datos, especialmente para aquellos usuarios no familiarizados o con nulo conocimiento en el lenguaje de consulta SQL; ya que podría reducir significativamente las barreras de entrada y mejorar la eficiencia de la recuperación de información en bases de datos.

El objetivo principal de este trabajo es diseñar, desarrollar y evaluar un sistema capaz de traducir consultas de lenguaje natural en consultas SQL de manera precisa y eficiente. Además, se busca proporcionar una justificación clara y detallada de las decisiones metodológicas adoptadas a lo largo del proceso de investigación.

La estructura del trabajo se organiza en varios capítulos, comenzando con una presentación sobre el tema que establece el contexto y los objetivos del proyecto. Posteriormente, se presenta una revisión exhaustiva del estado del arte en la generación automática de SQL, seguida de un marco teórico que sustenta los conceptos clave utilizados en el desarrollo del modelo. Luego, se analiza y diseña el modelo propuesto, detallando sus alcances, requerimientos funcionales, arquitectura y tecnologías involucradas. La implementación del modelo se describe en el siguiente capítulo, seguido de pruebas y validación para evaluar su rendimiento. Finalmente, se presentan las conclusiones obtenidas y se plantean posibles líneas de investigación futura que se realizará cuando el proyecto esté finalmente completado.

CAPÍTULO 1. Preámbulo

1.1 Situación problemática

SQL, también conocido como Structured Query Language, se originó en 1974. Donald Chamberlin y Raymond Boyce crearon un lenguaje para manipular y gestionar los datos almacenados en bases de datos relacionales. En su etapa inicial, el proyecto se conocía como "SEQUEL", que era un acrónimo de Structured English Query Language, lo que demostraba la intención de sus creadores de hacer que el lenguaje fuera similar al idioma inglés para que los usuarios pudieran leer las consultas de manera más natural y comprensible [1]. Posteriormente el nombre se cambió a SQL, el cual se convirtió en el lenguaje de consulta estándar para administrar y manipular bases de datos relacionales.

Aunque SQL puede leerse de forma intuitiva y puede ser muy útil para encontrar información valiosa, acceder al contenido de bases de datos relacionales requiere de preparación técnica. A medida que aumenta la necesidad de información más específica y detallada, las consultas pueden volverse más complejas y difíciles de manejar si no se cuenta con un conocimiento profundo del lenguaje y sobre la estructura de la base de datos en cuestión. Debido a esta limitación, los científicos de la computación han intentado darle solución a esta tarea de conversión de lenguaje natural a SQL, a la cual se denominó "*text to SQL*", que busca convertir el texto de una consulta en lenguaje natural a SQL.

Uno de los primeros enfoques propuestos para esta tarea fue el de Woods para el proyecto LUNAR [2]. LUNAR fue capaz de comprender consultas en lenguaje natural relacionadas con una base de datos que contenía análisis químicos de rocas lunares. Sin embargo, tanto el sistema de Woods como muchos de los sistemas desarrollados en los años posteriores tenían una limitación importante: estaban diseñados para trabajar con bases de datos específicas. Lo anterior hacía difícil, e incluso imposible, adaptarlos para trabajar con bases de datos externas.

Desde la aparición de LUNAR y hasta la actualidad, muchos de estos sistemas han sido desarrollados con el inglés en mente como idioma de consulta. Los pocos sistemas que han surgido con el español como idioma de consulta suelen ser diseñados para ser utilizados con una base de datos en específico [3], [4], lo cual limita el alcance del sistema y evita que este pueda ser utilizado con otras bases de datos. Esto nos indica el poco avance sobre la resolución de la tarea ya mencionada pero en el idioma español.

La comunidad de la inteligencia artificial ha enfocado sus esfuerzos en el estudio de la tarea "*text to SQL*" en el idioma inglés, debido a esto, para el español no se han presentado muchos avances en esta área a pesar de la revolución que ha supuesto el aprendizaje profundo en las tareas de generación de texto. En contraste tenemos a la comunidad de habla inglesa, la cual ha aprovechado los avances en el estado del arte del aprendizaje automático y ha alcanzado nuevos niveles de efectividad mediante distintos enfoques. Por ejemplo, tenemos el propuesto por Cai, que hizo uso de un enfoque basado en grafos al cual denominó SADGA (Structure-Aware Dual Graph Aggregation Network for Text-to-SQL)[5], o el propuesto por Mellah al usar un modelo de lenguaje pre-entrenado como T5 (Text-to-text transfer transformer) [6].

Con este trabajo terminal se pretende retomar la tarea de convertir consultas en lenguaje natural al lenguaje SQL para el idioma español. Se busca así promover el avance de la inteligencia artificial en la comunidad de habla hispana aprovechando las oportunidades no exploradas por otros investigadores, tales como la independencia de un modelo para realizar consultas en cualquier base de datos y el uso de nuevas técnicas de aprendizaje profundo. Esto permitirá a los usuarios hacer consultas de manera sencilla y accesible sin invertir mucho tiempo creando las consultas en SQL, y además ayudará a reducir la brecha con la comunidad de habla inglesa en este campo.

1.2 Justificación

La gestión de bases de datos (BD) es una tarea crítica en cualquier organización moderna para poder manipular y consultar las grandes cantidades de datos. Según un estudio realizado por IDC (International Data Corporation) en 2018, se espera que la cantidad de datos en el mundo alcance los 175 zettabytes para el año 2025 [7]. Sin embargo, la manipulación de estas BD se realiza típicamente utilizando el lenguaje SQL, lo que puede resultar en una actividad difícil y compleja. Incluso la simple tarea de consultar los datos de una persona dentro de la organización requeriría de alguien con conocimientos en SQL para lograr dicha consulta en la BD.

La idea anterior nos conduce a la necesidad crítica de contar con personal especializado en el manejo del lenguaje SQL para poder interactuar con los datos que se encuentran en la BD. Por esta razón, nuestra propuesta es desarrollar un modelo que pueda ser utilizado como parte de las herramientas de trabajo de los profesionales, de manera que contribuya a aumentar la eficiencia y el desempeño de aquellos que trabajen con consultas SQL.

Existen trabajos publicados para abordar esta tarea en el idioma español, los cuales se basan en el análisis de la estructura de la oración [4], [8], mientras que otros fuerzan al usuario a escribir su consulta con reglas preestablecidas [3], lo que impide la generación de consultas que no cumplan con estas y reduce la capacidad de expresividad del usuario. Además, algunos modelos están diseñados para trabajar con una sola base de datos [4], lo que imposibilita su uso en otras bases de datos.

Por otro lado, los modelos basados en tecnologías de vanguardia se centran principalmente en el idioma inglés [9]. Dichos modelos excluyen a la comunidad hispanohablante de tener acceso a modelos de inteligencia artificial para resolver esta problemática. Con la llegada de ChatGPT, un gran modelo de lenguaje (LLM por sus siglas en inglés) entrenado con más de 175 mil millones de parámetros [10], se han logrado importantes avances en distintas tareas que involucran el lenguaje natural. Y, aunque herramientas como ChatGPT han avanzado significativamente en distintas labores, incluyendo la de text-to-SQL [11] y su variante para el español, este problema aún no está resuelto, y muchas de las posibles rutas de investigación en este campo para el idioma español aún no han sido exploradas.

Esta propuesta no busca competir con herramientas como ChatGPT, sino aportar al campo de la generación de consultas SQL a partir de consultas en español, esperando que este trabajo pueda servir como una base sólida para futuras investigaciones en el área de la generación de consultas

SQL. Así que, el presente trabajo buscará implementar la combinación de técnicas de procesamiento de lenguaje natural (PLN) y aprendizaje automático (ML) para transformar consultas de lenguaje natural (en español) a consultas en SQL.

Se consideran problemas encontrados en los modelos existentes para el idioma español, por lo que, se busca generar un modelo versátil. Esto significa que el modelo debe ser capaz de procesar de manera efectiva una amplia variedad de preguntas y consultas complejas en diferentes contextos y dominios. Para lograr esto, se debe trabajar en la creación de un modelo lo suficientemente flexible y adaptable para manejar diferentes tipos de consultas y contextos.

Los usuarios potenciales de esta herramienta son empresas, organizaciones e individuos que requieren el acceso a información contenida en bases de datos. Los beneficios a los que podrán acceder incluyen el aumento de la productividad a la hora de trabajar con bases de datos SQL, gracias a la facilidad y rapidez con la que podrán realizar consultas a una base de datos relacional.

Se busca proveer una documentación clara y completa del modelo propuesto y su implementación. Esto incluirá la creación del documento actual y los próximos a realizar durante el proceso del trabajo terminal, igualmente una documentación técnica que permita a otros desarrolladores e investigadores entender y replicar el modelo propuesto.

En conclusión, el desarrollo de este trabajo presenta un gran desafío debido a la necesidad de diseñar e implementar un modelo de PLN y ML preciso y confiable. Además la elaboración de este proyecto formará parte de los avances para la investigación de esta tarea en el idioma español, lo que requiere de una rigurosa evaluación y prueba del prototipo resultante. En cuanto a la viabilidad del proyecto, se cuenta con el tiempo y recursos necesarios para llevar a cabo las diferentes etapas del desarrollo y evaluación del modelo.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar un modelo generativo de SQL a partir de consultas en español. El modelo podrá ser utilizado como parte de una herramienta que facilite a los usuarios el proceso de consulta de información en bases de datos relacionales. Esto se logrará a través de un enfoque que combina técnicas de procesamiento del lenguaje natural y aprendizaje automático, lo cual permitirá a los usuarios formular preguntas de manera más natural y obtener resultados precisos sin tener que invertir tanto tiempo en redactar su consulta en lenguaje SQL.

1.3.2 Objetivos específicos

- Realizar una investigación exhaustiva de las estructuras sintácticas y semánticas correctas de SQL y de cómo se relacionan con las expresiones naturales humanas.
- Construir un conjunto de ejemplos etiquetados para entrenar al modelo, incluyendo una gran cantidad de consultas en español, esquemas de bases de datos, y sus correspondientes representaciones en SQL.

- Seleccionar una arquitectura con la capacidad suficiente para capturar patrones complejos de dependencias entre palabras y partes de la frase en el texto en español y sus equivalentes en SQL.
- Crear e implementar un modelo de aprendizaje automático para convertir consultas escritas en español a consultas SQL.
- Asegurar la efectividad y corrección del modelo generado, evaluando y validando a través de pruebas exhaustivas.

1.4 Metodología

1.4.1 Metodología CRISP-ML

Se utiliza la metodología CRISP-ML(Q) debido a sus beneficios, ya que es muy recomendable para proyectos de ML [12]. La primera ventaja observada es que nos da un marco específico para el desarrollo del proyecto, lo que nos ayuda a evitar errores típicos y asegura una adecuada y efectiva implementación del modelo. Además, esta metodología nos permite trabajar de forma iterativa, lo que nos ayudará a mejorar continuamente nuestro modelo a medida que acumulamos nueva información o retroalimentación. Además, esta metodología destaca el valor de las pruebas y validaciones adecuadas, asegurándose que el modelo sea preciso y confiable antes de ser implementado en producción.

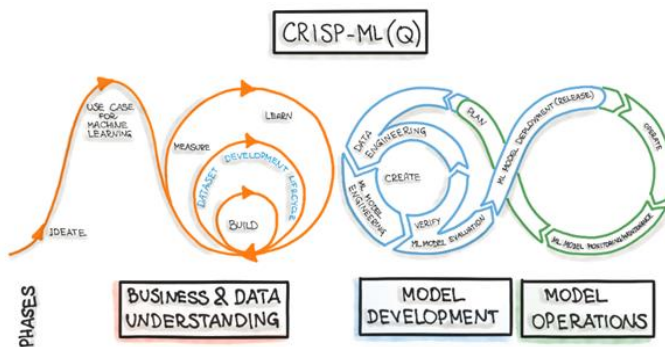
A continuación se presenta una breve descripción de las actividades que se realizarán en cada fase de la metodología:

- Fase 1: Comprender el problema. El objetivo de esta fase es identificar el problema que debe resolverse, que es el desarrollo de un modelo que convierta el lenguaje natural en SQL y permita a los usuarios ejecutar consultas de bases de datos sofisticadas. Se determinarán los objetivos y especificaciones de la herramienta, así como las dificultades o limitaciones tecnológicas que puedan surgir durante el desarrollo.
- Fase 2: Exploración de datos. Los datos necesarios para crear el modelo se recopilarán durante esta fase y se evaluarán. Se examinarán en profundidad las bases de datos accesibles, su arquitectura interna y las consultas más frecuentes que se realizan sobre ellas. Además, se recopilarán ejemplos de consultas en lenguaje natural para el entrenamiento del modelo de aprendizaje automático.
- Fase 3: Preparación de datos. Los datos necesarios para entrenar el modelo de aprendizaje automático se prepararán en esta etapa. Para que las consultas de lenguaje natural de ejemplo se puedan usar como entrada del modelo, se editarán y ordenarán.
- Fase 4: Modelamiento. El modelo de aprendizaje automático necesario para traducir consultas en lenguaje natural a SQL se desarrollará durante esta fase. El modelo se entrenará utilizando técnicas de procesamiento de lenguaje natural y aprendizaje automático utilizando las muestras de datos preparadas del paso anterior. Se evaluará el rendimiento del modelo y se realizarán los cambios necesarios.

- Fase 5: Evaluación. Esta etapa incluirá la evaluación del modelo terminado. La corrección y la fiabilidad del modelo para convertir consultas en lenguaje natural a SQL se probarán exhaustivamente. También se evaluará la usabilidad del modelo resultante.
- Fase 6: Implementación. Se realizaron las últimas pruebas antes de ser implementada y presentada ante el comité de evaluación.

A continuación, se muestran los diagramas simplificados que ilustran el funcionamiento de CRISP-ML:

(a)



(b)

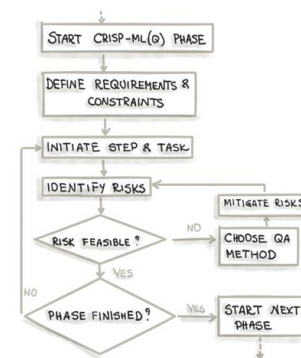


Fig 1.1. Diagramas simplificados que ilustran el funcionamiento de CRISP-ML. (a) Proceso del ciclo de vida del desarrollo del aprendizaje automático. (b) Enfoque CRISP-ML(Q) para asegurar la calidad para cada una de las fases [\[13\]](#).

CAPÍTULO 2. Estado del arte

La intersección entre los lenguajes de programación y el lenguaje humano ha sido un área de estudio prolífica en las últimas décadas. La conversión de lenguaje natural a SQL (Structured Query Language) representa un nicho específico en esta encrucijada. SQL, como lenguaje de consulta para la gestión y recuperación de datos en bases de datos relacionales, ha mantenido una estructura y sintaxis relativamente consistentes a lo largo de los años. Por otro lado, el lenguaje natural, rico y variado, se presenta como un desafío para ser traducido de manera precisa en instrucciones SQL.

En un sentido práctico, poder traducir consultas en lenguaje natural a SQL permite que usuarios sin conocimientos técnicos específicos puedan interactuar e interrogar bases de datos, además de también servir como herramienta de trabajo para los profesionales que se encarguen de la administración de bases de datos. Esto resulta en una mayor accesibilidad y usabilidad de los sistemas de información. En este contexto, la inteligencia artificial y, específicamente, el procesamiento del lenguaje natural (PLN) han sido recursos tecnológicos vitales.

Uno de los desafíos principales en la traducción de lenguaje natural a SQL es la ambigüedad del lenguaje humano. El mismo término o frase puede tener diferentes significados dependiendo del contexto en el que se utilice. Por lo tanto, los sistemas de PLN deben ser capaces de discernir la intención del usuario y traducir esa intención a un formato técnico, en este caso, consultas SQL.

A lo largo de los años, se han propuesto y desarrollado diferentes enfoques para abordar esta tarea. Inicialmente, los enfoques basados en reglas y gramáticas eran populares, donde las consultas en lenguaje natural se traducen a SQL a través de una serie de reglas predefinidas y mapeos de gramática. Aunque estos sistemas eran capaces de manejar consultas simples y estructuradas, a menudo luchaban con las consultas en lenguaje natural que se desviaban de las estructuras esperadas, tales como [4], [8] y [3] en el cual fuerzan al usuario a escribir su consulta con reglas preestablecidas.

Con el advenimiento de los modelos de aprendizaje profundo y, en particular, los modelos de atención y transformer, como BERT y GPT, la comunidad de investigación ha explorado enfoques basados en aprendizaje automático para la tarea. Estos modelos como podemos observar en [11] tienen la capacidad de manejar el lenguaje natural de una manera más flexible y generalizable, permitiendo traducciones más precisas y robustas en una variedad de contextos y dominios.

Se han propuesto también enfoques híbridos que combinan reglas lingüísticas y modelos de aprendizaje automático para mejorar la precisión y la robustez de la traducción del lenguaje natural a SQL. Estos sistemas buscan aprovechar las fortalezas de ambos enfoques, utilizando reglas y gramáticas para manejar casos claros y bien definidos, y modelos de aprendizaje automático para gestionar las ambigüedades y variabilidades del lenguaje natural; ejemplo de estos, tenemos a [12] y [13] que profundizan en el área del deep learning y las reglas lingüísticas.

La interpretación de lenguaje natural para generar consultas SQL coherentes y precisas se adentra profundamente en el campo de la semántica, donde se busca entender no sólo las palabras individuales, sino también el significado que generan en conjunto. Los modelos más

recientes de traducción de lenguaje natural a SQL, buscan comprender las implicaciones semánticas de las consultas, incluyendo las subconsultas y las referencias a múltiples tablas en una base de datos. Esto se complejiza aún más cuando las bases de datos tienen esquemas complejos y diversas relaciones entre sus tablas [16].

Por otro lado, a pesar de los avances significativos en este campo, la traducción de lenguaje natural a SQL sigue siendo un área activa de investigación y desarrollo. Los desafíos persistentes, como la gestión de la ambigüedad, la interpretación de referencias anafóricas, y la comprensión del contexto global de las consultas, continúan impulsando nuevas investigaciones y enfoques en esta área fascinante [16].

En el futuro, se espera que los sistemas de traducción de lenguaje natural a SQL se vuelvan más robustos y capaces de manejar consultas cada vez más complejas, incluyendo la capacidad de gestionar consultas más complejas y condiciones de filtrado avanzadas. Además, los modelos futuros también podrían tener la capacidad de aprender y adaptarse a las especificidades del dominio en el que están siendo utilizados, permitiendo una mayor personalización y precisión en las traducciones generadas [14].

2.1 Revisión de investigaciones anteriores en generación automática de SQL

A continuación, se presentará un breve resumen de investigaciones y trabajos relacionados con la solución a la tarea de "Text to SQL". En algunos de estos *papers*, se abordan diversas estrategias y enfoques para la traducción de texto natural a consultas SQL, y se mostrará la arquitectura que han implementado para abordar este desafío. Estos estudios proporcionan una visión general de cómo se ha abordado este problema y ofrecen ideas clave sobre cómo se ha avanzado en la generación automática de consultas SQL a partir de lenguaje natural.

Para el idioma Inglés:

- *Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning* [17].

Es un sistema avanzado que utiliza inteligencia artificial para convertir preguntas en lenguaje natural en consultas SQL, el lenguaje de las bases de datos. Este enfoque facilita a los usuarios la obtención de información de bases de datos sin necesidad de entender el complejo lenguaje SQL. Utilizando técnicas de aprendizaje profundo y recompensas de ejecución de consultas, Seq2SQL mejora la precisión en la generación de consultas. Además, se presenta WikiSQL, una base de datos de preguntas y consultas SQL que es significativamente más grande que otras disponibles.

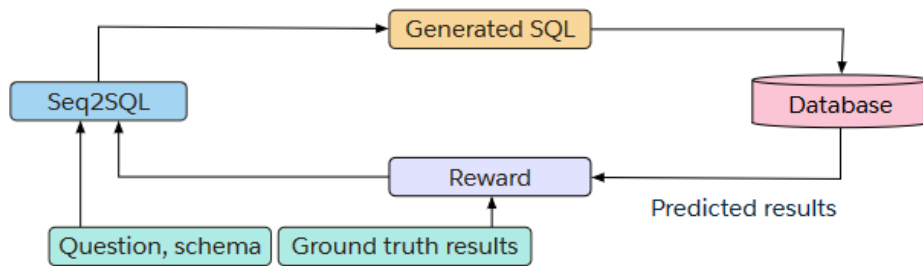


Fig 2.1. Diagrama del funcionamiento Seq2SQL para la generación de una consulta en SQL y su enfoque de aprendizaje por refuerzo. [17]

- *SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning* [18].

SQLNet es una propuesta para abordar el problema de sintetizar consultas SQL a partir de descripciones en lenguaje natural (NL2SQL). SQLNet utiliza un enfoque basado en esquemas, evitando la estructura secuencia a secuencia cuando el orden no es relevante. Este enfoque utiliza un esquema que contiene un grafo de dependencias, permitiendo que una predicción se realice considerando solo las predicciones anteriores en las que depende. Además, SQLNet introduce un modelo de secuencia a conjunto y un mecanismo de atención de columnas para sintetizar la consulta basándose en el esquema.

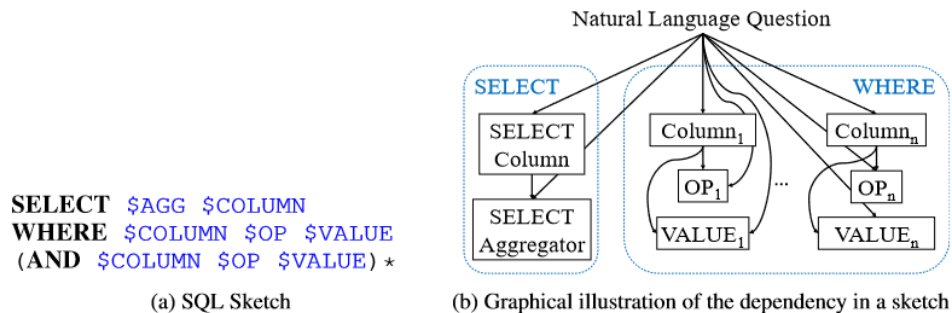


Fig 2.2. Diagrama que muestra la propuesta hecha en SQLNet para el tratamiento de los esquemas SQL y su enfoque de grafo de dependencias [18].

- *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task* [19].

Es un conjunto de datos grande y complejo que se centra en la tarea de la interpretación semántica y la generación de consultas SQL a partir de preguntas en lenguaje natural.

Spider se distingue de otros conjuntos de datos de interpretación semántica por la presencia de consultas SQL complejas y la variación de bases de datos entre los conjuntos de entrenamiento y prueba.

Algunos detalles específicos de este conjunto de datos son los siguientes:

- Número total de preguntas: 10,181
- Número total de consultas SQL únicas: 5,693
- Bases de datos cubiertas: 200

- Dominios cubiertos: 138 [19]
- *ValueNet: A Natural Language-to-SQL System that Learns from Database Information [20].*

ValueNet es un sistema NL-to-SQL. La principal característica distintiva de ValueNet es su capacidad para incorporar valores específicos mencionados en las preguntas de los usuarios en el proceso de generación de consultas SQL. Esto permite a los usuarios realizar consultas complejas en bases de datos sin tener que conocer SQL o la estructura subyacente de la base de datos. ValueNet utiliza técnicas avanzadas de aprendizaje automático y propone dos enfoques, ValueNet light y ValueNet, ambos demostrando resultados destacados en un desafío llamado Spider. La implementación incluye un procesamiento previo de las preguntas, un codificador-decodificador basado en *transformers* y un paso de posprocesamiento para transformar las representaciones intermedias en consultas SQL ejecutables.

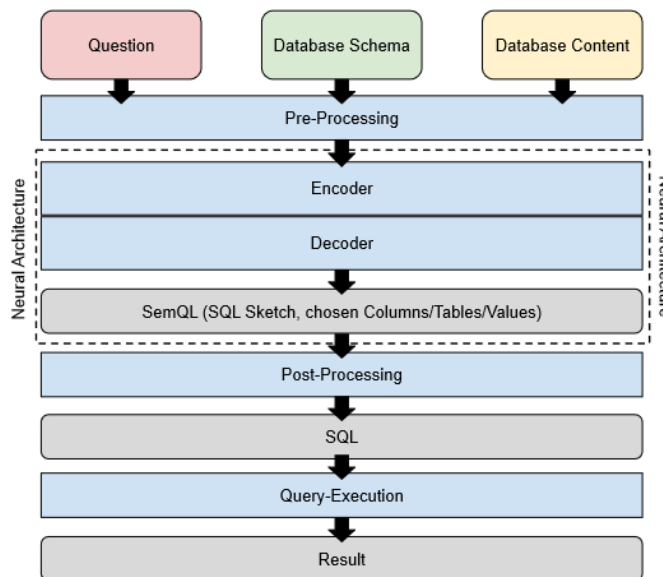


Fig 2.3. Diagrama que representa una descripción general de la arquitectura propuesta ValueNet [20].

- *RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers [21].*

Es un enfoque para mejorar la capacidad de los modelos de análisis semántico para traducir preguntas en lenguaje natural a consultas SQL, especialmente cuando se enfrentan a esquemas de base de datos no vistos previamente. El desafío principal radica en codificar las relaciones de la base de datos de manera accesible y alinear las columnas de la base de datos con sus menciones en una pregunta. El marco propuesto utiliza un mecanismo de atención propio, llamado relación-aware self-attention, para abordar la codificación del esquema, la vinculación del esquema y la representación de características.

- SADGA [5].

SADGA aborda el desafío de Text-to-SQL, que consiste en traducir preguntas en lenguaje natural a consultas SQL, especialmente en escenarios con esquemas de bases de datos no vistos anteriormente. La clave de SADGA radica en su enfoque estructural, utilizando la estructura de grafo para unificar la codificación de la pregunta y el esquema de la base de datos. Introduce un método de agregación estructural que incluye Enlace Global del Grafo, Enlace Local del Grafo y un Mecanismo de Agregación de Doble Grafo. Este enfoque logró el tercer lugar en el desafiante conjunto de datos Text-to-SQL Spider, destacando su eficacia en la generalización entre dominios en el ámbito de Text-to-SQL.

- CatSQL [22].

El estudio presenta CatSQL, aborda limitaciones comunes en enfoques existentes. CatSQL utiliza un esquema de plantilla que se integra con modelos de aprendizaje profundo, eliminando la generación de palabras clave y logrando mayor precisión y velocidad en comparación con enfoques secuencia a secuencia, destaca por su versatilidad al aprovechar valores completados y por la introducción de la técnica de Corrección Semántica, que utiliza conocimiento de dominio para mejorar la precisión de las consultas SQL generadas. En evaluaciones extensas, CatSQL supera significativamente soluciones anteriores.

- ChatGPT [11].

En el ámbito de desarrollo de ChatGPT, la transformación de lenguaje natural a consultas SQL se realiza mediante la integración de técnicas avanzadas de procesamiento del lenguaje natural (NLP). ChatGPT utiliza modelos NLP para descomponer las consultas, identificar entidades relevantes como nombres de tablas y columnas, y construir una estructura sintáctica coherente. Posteriormente, se genera la consulta SQL teniendo en cuenta la relación entre las entidades identificadas. Este enfoque busca mejorar la interacción usuario-máquina al permitir que los usuarios formulen consultas SQL de manera más intuitiva y conversacional. La validación y optimización de la consulta se realizan para garantizar la corrección sintáctica y semántica antes de su ejecución en la base de datos.

Para el idioma español:

- Traductor de consultas del lenguaje natural a SQL de Bonilla [4].

Este proyecto utiliza diversas técnicas de procesamiento del lenguaje natural para el manejo de la información. En este proceso, se emplea un diccionario de sinónimos que actúa como una base de datos para llevar a cabo un análisis léxico, sintáctico y semántico de la información. Esta metodología permite un tratamiento exhaustivo de los datos. Además, se implementa un sistema de gestión de base de datos que se ejecuta en conjunto con el programa, y la respuesta se devuelve a través de la interfaz correspondiente.

- LNE2SQL[23].

La interfaz LNE2SQL se centra en la transformación de lenguaje natural a consultas SQL, utilizando un dominio de base de datos de WordNet. El proceso implica el uso de diversas técnicas de procesamiento de lenguaje natural, complementando la base de datos para

abordar consultas más complejas. El modelo se basa en una base de datos que incluye un diccionario de sinónimos y metadatos. El diccionario de sinónimos facilita la identificación de palabras similares proporcionadas por el usuario, mientras que el diccionario de metadatos ofrece información sobre tablas, número de columnas, nombres, llaves foráneas y primarias.

Cada nombre de columna y tabla se asocia con los sustantivos presentes en su descripción y los sinónimos correspondientes en el diccionario de sinónimos. Un analizador léxico verifica la estructura de las oraciones, incluyendo elementos básicos como sustantivos, verbos, preposiciones y artículos. Se utilizan palabras clave en la oración como pautas para identificar los elementos. El análisis semántico tiene como objetivo capturar las partes que dan sentido a la consulta.

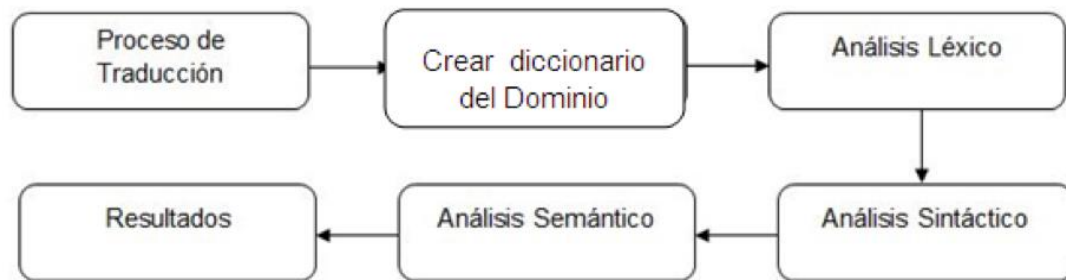


Fig 2.4. Diagrama que representa una descripción general de la arquitectura propuesta ValueNet [\[23\]](#).

- AlcoNQL [\[3\]](#).

El proyecto AlqoSQL se centra en desarrollar una herramienta que transforma sentencias en lenguaje natural en consultas SQL SELECT, posibilitando a usuarios sin conocimientos técnicos realizar consultas a bases de datos sin la necesidad de comprender el lenguaje técnico asociado. Este proyecto se divide en varios hitos, incluyendo la creación de la herramienta de transformación, el establecimiento de una base de datos interactiva, la implementación de una comunicación efectiva entre la herramienta y la base de datos, y la creación de una interfaz gráfica de usuario que integra todas las funcionalidades para mejorar la experiencia del usuario en el proceso de formulación y ejecución de consultas. La iniciativa tiene como objetivo eliminar las barreras entre usuarios no técnicos y bases de datos, facilitando la realización de consultas mediante un enfoque más accesible y natural.

CAPÍTULO 3. Marco teórico

3.1 Lenguaje SQL

3.1.1 Estructura básica de una consulta SQL

El lenguaje principal para realizar consultas y manipular datos en tablas se llama Structured Query Language, comúnmente abreviado como SQL. Fue estandarizado por ANSI (American National Standards Institute) e ISO (International Organization for Standardization) [24].

SQL es un lenguaje descriptivo, ya que las declaraciones describen el resultado deseado en lugar de los pasos de cálculo necesarios. Las consultas SQL siguen un patrón básico, por lo general tienen la estructura SELECT-FROM-WHERE, y siempre generan una tabla como resultado. En la Figura 3.1 podemos ver un ejemplo de cómo formular la consulta “Selecciona el nombre de los empleados que viven en Kent” en el lenguaje SQL, todo esto partiendo de una base de datos que solo cuenta con la tabla “Empleado”, tal como puede verse en la parte superior de la Figura 3.1:

Empleado

Número de empleado	Nombre	Ciudad
E19	Stewart	Stow
E4	Bell	Kent
E19	Murphy	Kent
E7	Howard	Cleveland

Ejemplo de consulta

"Selecciona el nombre de los empleados que viven en Kent"

```
SELECT      Nombre
FROM        Empleado
WHERE       Ciudad = "Kent"
```

Resultado de la tabla:

Nombre
Bell
Murphy

Fig 3.1. Formulando una consulta en SQL [24].

Como puede verse en la consulta mostrada en la Figura 1.1, el lenguaje SQL requiere que se identifiquen tres elementos principales, los cuales son los atributos a obtener (cláusula SELECT), el nombre de la tabla donde se encuentran los atributos de interés (cláusula FROM), y la condición para filtrar los resultados en la tabla (cláusula WHERE). Los elementos mencionados anteriormente, según la descripción que se da (sin incluir elementos de mayor complejidad, como lo pueden ser los agregadores, unión de tablas, condiciones con más de una comparación lógica, etc.), serán los que consideraremos como parte de la “estructura básica” a lo largo de este trabajo.

Con esto en mente, un usuario de SQL podría leer la consulta como "SELECT el atributo 'Nombre' FROM la tabla 'EMPLEADO' WHERE la ciudad es Kent". En este ejemplo, la consulta arrojaría una tabla de resultados con los nombres Bell y Murphy, tal como se esperaba [24].

Ahora que se han ejemplificado los elementos de la consulta básica, procederemos a profundizar en otras cláusulas que pueden utilizarse como parte de una consulta SQL.

3.1.2 Componentes de una consulta SQL

El lenguaje de consulta estructurado (SQL) es una herramienta fundamental en la gestión de bases de datos relacionales, y su dominio es crucial para quienes trabajan en el ámbito de la administración y manipulación de datos [24]. Una de las características distintivas del SQL es su capacidad para emplear una variedad de componentes en la formulación de consultas, lo que permite realizar una amplia gama de operaciones y análisis de datos [25]. Entender estos componentes es fundamental para diferenciar los tipos de consultas y niveles de operaciones que pueden llevarse a cabo en una base de datos relacional.

A continuación, se muestran en detalle los componentes principales de una consulta SQL extraídos de [25] junto con ejemplos ilustrativos:

1. Cláusula SELECT

La cláusula SELECT se utiliza para especificar las columnas que se mostrarán en los resultados de la consulta. Por ejemplo:

```
SELECT nombre, edad, ciudad FROM clientes;
```

Este ejemplo selecciona las columnas "nombre", "edad" y "ciudad" de la tabla "clientes".

2. Cláusula FROM

La cláusula FROM indica las tablas de las que se extraerán los datos. Por ejemplo:

```
SELECT * FROM productos;
```

Esta consulta selecciona todas las columnas de la tabla "productos".

3. Cláusula WHERE

La cláusula WHERE define las condiciones que deben cumplir las filas para ser incluidas en los resultados. Por ejemplo:

```
SELECT * FROM empleados WHERE departamento = 'Ventas';
```

Esta consulta selecciona todas las columnas de la tabla "empleados" donde el valor de la columna "departamento" es 'Ventas'.

4. Cláusula ORDER BY

La cláusula ORDER BY ordena los resultados según una o más columnas. Por ejemplo:

```
SELECT nombre, salario FROM empleados ORDER BY salario DESC;
```

Esta consulta selecciona las columnas "nombre" y "salario" de la tabla "empleados" y ordena los resultados en orden descendente según el salario.

5. Cláusula GROUP BY

La cláusula GROUP BY agrupa filas con valores idénticos en una o más columnas. Por ejemplo:

```
SELECT departamento, COUNT(*) FROM empleados GROUP BY departamento;
```

Esta consulta cuenta el número de empleados en cada departamento y muestra el resultado agrupado por departamento.

6. Funciones Agregadoras

Las funciones agregadoras, como SUM, AVG, COUNT, MAX y MIN, permiten realizar cálculos en conjuntos de filas. Por ejemplo:

```
SELECT MAX(salario) FROM empleados;
```

Esta consulta devuelve el salario máximo de la tabla "empleados".

7. Cláusulas JOIN

Las cláusulas JOIN permiten combinar datos de varias tablas basadas en una condición de relación. Por ejemplo, en la consulta:

```
SELECT empleados.nombre, departamentos.nombre  
FROM empleados  
JOIN departamentos ON empleados.departamento_id = departamentos.id;
```

Se combinan los datos de las tablas "empleados" y "departamentos" en aquellas columnas que donde se cumple la igualdad entre las columnas "departamento_id" y "id".

Después de explorar los aspectos esenciales de la formulación de una consulta SQL, se explicarán algunos temas relacionados en el ámbito del aprendizaje automático.

3.2 Aprendizaje de máquina

3.2.1 Introducción al aprendizaje de máquina

Es un campo de la inteligencia artificial (IA) que se centra en el desarrollo de sistemas capaces de aprender y mejorar su rendimiento sin intervención humana directa. La esencia del aprendizaje automático radica en la capacidad de las máquinas para extraer patrones y conocimientos a partir

de datos, y utilizar esa información para tomar decisiones o realizar tareas específicas. Una de las bondades del aprendizaje automático es la manera de ser programados de manera explícita, estos algoritmos aprenden mediante los datos e identificación de patrones. Estos algoritmos pueden generalizar a partir de ejemplos previos y aplicar ese conocimiento para abordar nuevas situaciones o realizar predicciones [26].

Estos algoritmos se clasifican en tres secciones principales:

Aprendizaje Supervisado: El algoritmo se entrena utilizando un conjunto de datos etiquetado, donde la entrada y la salida deseada están claramente definidas. El objetivo es que el modelo haga predicciones o clasificaciones basadas en ejemplos conocidos [26].

Aprendizaje No Supervisado: El algoritmo se entrena con datos no etiquetados, y el sistema debe descubrir patrones y estructuras por sí mismo. Esto se utiliza comúnmente para la segmentación y la reducción de dimensionalidad [26].

Aprendizaje por Reforzamiento: El algoritmo aprende a través de la interacción con un entorno. Se refuerza positivamente por decisiones correctas y negativamente por decisiones incorrectas, lo que lleva a un aprendizaje continuo y adaptativo [26].

Con esta comprensión del aprendizaje de máquina, se explorará cómo se aplica el reconocimiento de patrones para extraer información valiosa de conjuntos de datos en diversas aplicaciones.

5.2.2 Reconocimiento de patrones

El reconocimiento de patrones es un campo cuyo objetivo principal es extraer de manera óptima patrones basados en un conjunto de datos proporcionado con el fin de distinguir una clase de las demás. La aplicación del reconocimiento de patrones se encuentra en todas partes, incluyendo la categorización de enfermedades, predicción de tasas de supervivencia para pacientes con enfermedades específicas, verificación de huellas dactilares, reconocimiento facial, discriminación de iris, discriminación de formas de cromosomas, reconocimiento óptico de caracteres, discriminación de texturas, reconocimiento de voz, entre otros. El diseño de un sistema de reconocimiento de patrones debe tener en cuenta el dominio de aplicación, ya que no existe un sistema universalmente óptimo [27].

Los componentes básicos de un sistema de reconocimiento de patrones son el preprocesamiento, la extracción de características y la clasificación.

Preprocesamiento:

El preprocesamiento incluye tareas como la tokenización (división del texto en unidades más pequeñas, como palabras o símbolos), eliminación de stopwords (palabras comunes que se consideran irrelevantes para el análisis, como "el", "un", "y") y lematización (reducción de las palabras a su forma base o lema, por ejemplo, "corriendo" se convierte en "correr"), que preparan el texto para el análisis posterior [27].

Extracción de características

La extracción de características consiste en identificar atributos relevantes en el texto que puedan utilizarse para distinguir entre clases, como la frecuencia de palabras o la presencia de ciertos términos clave [27].

Clasificación:

Finalmente, la clasificación implica asignar una etiqueta o categoría a un texto en función de las características identificadas, utilizando técnicas como la clasificación binaria, la clasificación multinomial o el aprendizaje profundo [27].

Con lo anterior, ahora se explicará la inducción de reglas, un componente esencial en el aprendizaje automático que permite descubrir patrones significativos en conjuntos de datos mediante la construcción de reglas lógicas

3.2.3 Inducción de reglas

La inducción por reglas es un pilar fundamental en el ámbito de la Inteligencia Artificial, especialmente en el aprendizaje automático. Se trata de un enfoque que se concentra en descubrir patrones significativos en conjuntos de datos mediante la construcción de reglas lógicas o condiciones que representan relaciones y regularidades presentes en la información analizada [28].

En esencia, la inducción por reglas implica la capacidad de los sistemas para aprender dichas reglas a partir de ejemplos y datos disponibles, permitiéndoles descubrir conexiones y tendencias a esos datos. Este método posibilita la generación de reglas lógicas comprensibles que se utilizan para predecir resultados, clasificar información o proporcionar explicaciones sobre cómo el sistema ha llegado a una conclusión o decisión específica [28].

En el contexto del aprendizaje de reglas de clasificación, el objetivo principal es encontrar un conjunto de reglas que permitan predecir o clasificar nuevas instancias, es decir, ejemplos que no han sido presentados previamente al modelo. Esto implica identificar patrones en los datos que permitan generalizar el conocimiento adquirido durante el proceso de entrenamiento y aplicarlo de manera efectiva a nuevas situaciones [29].

Siguiendo con el tema, se presentará el Algoritmo Quasi-Optimal (AQ), una herramienta crucial en el ámbito del aprendizaje automático que busca encontrar reglas de clasificación óptimas mediante un enfoque adaptativo y eficiente.

3.2.4 Algoritmo Quasi-Optimal (AQ)

El algoritmo AQ es un clasificador de aprendizaje automático que se fundamenta en un enfoque basado en reglas para la clasificación de datos. Este enfoque implica la generación de reglas explícitas que describen cómo se deben clasificar los diferentes ejemplos en función de los valores de sus atributos. Durante el proceso de entrenamiento, utilizando conjuntos de datos etiquetados en un entorno supervisado, el algoritmo AQ analiza estos datos para identificar patrones en los valores de los atributos que están correlacionados con las etiquetas de clase [30].

Utilizando estos patrones, AQ genera un conjunto de reglas fácilmente interpretables. Estas reglas pueden ser examinadas por los usuarios para comprender intuitivamente cómo se realiza la clasificación. Esta transparencia es crucial en muchos contextos donde es fundamental entender cómo se toman las decisiones de clasificación.

Funcionamiento

El algoritmo AQ comienza seleccionando dos conjuntos de eventos no vacíos: uno de eventos positivos y otro de eventos negativos, donde al menos un evento positivo y uno negativo no son ambiguos.

Luego, se crea una lista de eventos positivos aún no cubiertos, denominada P' . El algoritmo procede iterativamente hasta que todos los eventos positivos han sido cubiertos.

En cada iteración, se elige aleatoriamente un evento positivo de P' , llamado "semilla", y se genera una "estrella" para este ejemplo. El resultado de la estrella es una regla que generaliza la semilla y no cubre ninguno de los negativos (TF).

Durante la generación de estrellas, se utiliza una función de evaluación lexicográfica (LEF) para evaluar las reglas. Luego, se eliminan todos los eventos cubiertos por la regla r de la lista P' . La regla r está garantizada para cubrir al menos la semilla, pero podría cubrir muchos, si no todos, los eventos en P' . Finalmente, la regla r se agrega a la lista de reglas R para su inclusión en la respuesta final [30].

En la Figura 3.2 se presenta el pseudocódigo que describe el funcionamiento del algoritmo, sintetizando los pasos previamente mencionados.

Algorithm 1 High Level AQ Algorithm

Require: $|P| > 0$ AND $|N| > 0$

1: $P' \leftarrow P$; $R = 0$

2: **while** $|P'| > 1$ **do**

3: Select random p from P'

4: $r \leftarrow STAR(p, N, LEF, maxstar)$

5: $P' \leftarrow P' - [P' \cap r]$

6: $R \leftarrow R + r$

7: **end while**

Fig 3.2. Algoritmo AQ. Extraído de [30]

Generación de estrella

La generación de estrellas implica tomar una "semilla" y crear todas las combinaciones posibles de reglas de diferentes longitudes. Cada regla se evalúa para garantizar que cubra la semilla pero no cubra ningún ejemplo negativo, mientras que también se busca maximizar la cobertura de

ejemplos positivos. La regla seleccionada será aquella que cubra la mayor cantidad de ejemplos positivos. En caso de empate, se prioriza la regla más corta en longitud de atributos, y si persiste el empate, se elige una regla al azar. Este proceso se repite iterativamente para todas las semillas seleccionadas hasta que todos los ejemplos positivos estén cubiertos.

El proceso de generación de la "estrella" se detalla paso a paso en la figura 3.2 mediante pseudocódigo.

Algorithm 2 Star Generation Algorithm

```

1:  $r = 0$ 
2: for all  $n$  in  $N$  do
3:    $r' \leftarrow p \div n$ 
4:    $r'' \leftarrow r' \cup r$ 
5:    $r'' \leftarrow LEF(r'', maxstar)$ 
6:   if  $\{Mode = PD\}$  then
7:     if  $q(r'') - q(r) > minq$  then
8:        $r = r''$ 
9:     end if
10:  else
11:     $r = r''$ 
12:  end if
13: end for

```

Fig 3.3. Algoritmo de generación de estrella. Extraído de [30].

Ejemplo

Las figuras 1.4 a 1.8 detallan paso a paso el funcionamiento del algoritmo AQ. El ejemplo presentado en esta sección es de elaboración propia, y el conjunto de datos utilizado fue recuperado de [8]. Como resultado de esta implementación, se obtiene una regla general que cubre el problema de clasificación correspondiente.

En la figura 1.4 se recopila el conjunto de datos y se organiza según su clase. En este caso, se trata de un conjunto de datos para determinar si un globo debe inflarse. Las instancias positivas se representan con el color verde y las instancias negativas con el color rojo.

COLOR	SIZE	ACT	AGE	INFLATED?
YELLOW	SMALL	STRETCH	ADULT	T
YELLOW	SMALL	STRETCH	ADULT	T
YELLOW	LARGE	STRETCH	ADULT	T
YELLOW	LARGE	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	ADULT	T
PURPLE	SMALL	STRETCH	ADULT	T
PURPLE	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	CHILD	F
PURPLE	LARGE	DIP	ADULT	F
PURPLE	LARGE	DIP	CHILD	F
YELLOW	SMALL	STRETCH	CHILD	F
YELLOW	SMALL	DIP	ADULT	F
YELLOW	SMALL	DIP	CHILD	F
YELLOW	LARGE	STRETCH	CHILD	F
YELLOW	LARGE	DIP	ADULT	F
YELLOW	LARGE	DIP	CHILD	F
PURPLE	SMALL	STRETCH	CHILD	F
PURPLE	SMALL	DIP	ADULT	F
PURPLE	SMALL	DIP	CHILD	F

Fig 3.4. Paso 0: Cargar el conjunto de datos y dividirlo en 2 clases

En la figura 1.5, se elige aleatoriamente una "semilla" de la clase positiva y se desarrolla su "estrella". Esto implica generar todas las combinaciones posibles de diferentes longitudes. Luego, se evalúan las reglas de la "estrella" para asegurar que no cubran la clase negativa. Aquellas reglas que sí lo hacen son descartadas, como se muestra en la imagen: las reglas descartadas están marcadas en rojo y las aceptadas en verde. Por otro lado, entre las reglas aceptadas se aplican ciertos criterios para seleccionar la mejor. Primero, se evalúa cuántas instancias de la clase positiva cubre cada regla. En caso de empate, se elige la regla más corta, y si el empate persiste, se selecciona una aleatoriamente. En la parte inferior de la imagen se encuentra la regla seleccionada.

COLOR	SIZE	ACT	AGE	INFLATED?
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	DIP	ADULT	T
PURPLE	LARGE	STRETCH	ADULT	T
PURPLE	SMALL	STRETCH	ADULT	T
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	CHILD	F
PURPLE	LARGE	DIP	ADULT	F
PURPLE	LARGE	DIP	CHILD	F
YELLOW	SMALL	STRETCH	CHILD	F
YELLOW	SMALL	DIP	ADULT	F
YELLOW	SMALL	DIP	CHILD	F
YELLOW	LARGE	STRETCH	CHILD	F
YELLOW	LARGE	DIP	ADULT	F
YELLOW	LARGE	DIP	CHILD	F
PURPLE	SMALL	STRETCH	CHILD	F
PURPLE	SMALL	DIP	ADULT	F
PURPLE	SMALL	DIP	CHILD	F

SEMILLA					Regla				No. cubrimientos
PURPLE	LARGE	STRETCH	ADULT		PURPLE	STRETCH	ADULT		2
PURPLE	LARGE	STRETCH	ADULT		LARGE	STRETCH	ADULT		1
PURPLE	LARGE	STRETCH	ADULT						
PURPLE	LARGE	STRETCH							
PURPLE	LARGE	ADULT							
PURPLE	STRETCH	ADULT							
LARGE	STRETCH	ADULT							
PURPLE	LARGE								
PURPLE	STRETCH								
PURPLE	ADULT								
LARGE	STRETCH								
LARGE	ADULT								
STRETCH	ADULT								
PURPLE	LARGE								
PURPLE	STRETCH								
PURPLE	ADULT								
LARGE	STRETCH								
LARGE	ADULT								
STRETCH	ADULT								
PURPLE	LARGE								
PURPLE	STRETCH								
PURPLE	ADULT								
LARGE	STRETCH								
LARGE	ADULT								
STRETCH	ADULT								

Reglas				
[PURPLE, STRETCH, ADULT]				

Fig 3.5. Obtención de primera regla

En la figura 1.6, se repiten los procedimientos presentados en la figura 1.5. Esto implica seleccionar una "semilla" al azar, generar su "estrella", evaluarla respecto a la clase negativa y,

además, maximizar la cobertura de ejemplos positivos. Luego, se elige la regla más adecuada según los criterios de selección establecidos.

COLOR	SIZE	ACT	AGE	INFLATED?
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	DIP	ADULT	T
PURPLE	LARGE	STRETCH	ADULT	T
PURPLE	SMALL	STRETCH	ADULT	T
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	CHILD	F
PURPLE	LARGE	DIP	ADULT	F
PURPLE	LARGE	DIP	CHILD	F
YELLOW	SMALL	STRETCH	CHILD	F
YELLOW	SMALL	DIP	ADULT	F
YELLOW	SMALL	DIP	CHILD	F
YELLOW	LARGE	STRETCH	CHILD	F
YELLOW	LARGE	DIP	ADULT	F
YELLOW	LARGE	DIP	CHILD	F
PURPLE	SMALL	STRETCH	CHILD	F
PURPLE	SMALL	DIP	ADULT	F
PURPLE	SMALL	DIP	CHILD	F

REGLAS				
[PURPLE, STRETCH, ADULT], [PURPLE, LARGE, DIP, ADULT]				

SEMILLA			
PURPLE	LARGE	DIP	ADULT

PURPLE	LARGE	DIP	ADULT
PURPLE	LARGE	ADULT	
PURPLE	DIP	ADULT	
LARGE	DIP	ADULT	

PURPLE	LARGE
PURPLE	DIP
PURPLE	ADULT
LARGE	DIP
LARGE	ADULT
DIP	ADULT

PURPLE
LARGE
DIP
ADULT

Criterios de selección de reglas:
Longitud más corta, Número de instancias cubiertas, Selección aleatoria

Regla				Cubrimiento
PURPLE	LARGE	DIP	ADULT	1

[PURPLE, LARGE, DIP, ADULT]

Fig 3.6. Obtención de segunda regla

En la figura 1.7, se repiten los procedimientos presentados en la figura 1.5. Esto implica seleccionar una “semilla” al azar, generar su estrella, evaluarla respecto a la clase negativa y, además, maximizar la cobertura de ejemplos positivos. Luego, se elige la regla más adecuada según los criterios de selección establecidos.

COLOR	SIZE	ACT	AGE	INFLATED?
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	DIP	ADULT	T
PURPLE	LARGE	STRETCH	ADULT	T
PURPLE	SMALL	STRETCH	ADULT	T
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	CHILD	F
PURPLE	LARGE	DIP	ADULT	F
PURPLE	LARGE	DIP	CHILD	F
YELLOW	SMALL	STRETCH	CHILD	F
YELLOW	SMALL	DIP	ADULT	F
YELLOW	SMALL	DIP	CHILD	F
YELLOW	LARGE	STRETCH	CHILD	F
YELLOW	LARGE	DIP	ADULT	F
YELLOW	LARGE	DIP	CHILD	F
PURPLE	SMALL	STRETCH	CHILD	F
PURPLE	SMALL	DIP	ADULT	F
PURPLE	SMALL	DIP	CHILD	F

REGLAS				
[PURPLE, STRETCH, ADULT], [PURPLE, LARGE, DIP, ADULT], [STRETCH, ADULT]				

SEMILLA			
YELLOW	SMALL	STRETCH	ADULT

YELLOW	SMALL	STRETCH	ADULT
YELLOW	SMALL	ADULT	
YELLOW	STRETCH	ADULT	
SMALL	STRETCH	ADULT	

YELLOW	SMALL
YELLOW	STRETCH
YELLOW	ADULT
SMALL	STRETCH
SMALL	ADULT
STRETCH	ADULT

YELLOW
SMALL
STRETCH

ADULT

Criterios de selección de reglas:
Longitud más corta, Número de instancias cubiertas, Selección aleatoria

Regla				No. cubrimientos
STRETCH	ADULT			2
YELLOW	STRETCH	ADULT		2
	STRETCH	ADULT		2
YELLOW	SMALL	STRETCH	ADULT	1

Fig 3.7. Obtención de tercera regla

En la figura 1.8, el algoritmo llega a su conclusión. Esto se debe a que las reglas logran abarcar toda la clase positiva, asegurando que todos los ejemplos de dicha clase hayan sido cubiertos.

YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	DIP	ADULT	T
PURPLE	LARGE	STRETCH	ADULT	T
PURPLE	SMALL	STRETCH	ADULT	T
YELLOW	SMALL	STRETCH	ADULT	T
PURPLE	LARGE	STRETCH	CHILD	F
PURPLE	LARGE	DIP	ADULT	F
PURPLE	LARGE	DIP	CHILD	F
YELLOW	SMALL	STRETCH	CHILD	F
YELLOW	SMALL	DIP	ADULT	F
YELLOW	SMALL	DIP	CHILD	F
YELLOW	LARGE	STRETCH	CHILD	F
YELLOW	LARGE	DIP	ADULT	F
YELLOW	LARGE	DIP	CHILD	F
PURPLE	SMALL	STRETCH	CHILD	F
PURPLE	SMALL	DIP	ADULT	F
PURPLE	SMALL	DIP	CHILD	F

Todos los ejemplos fueron cubiertos

Fig 3.8. Visualización de ejemplos positivos completamente abarcados

A continuación, se explicarán algunos conceptos esenciales del Procesamiento de Lenguaje Natural; un campo que se encarga del estudio y desarrollo de técnicas para permitir que las computadoras comprendan, interpreten y generen lenguaje humano de manera efectiva.

3.3 Procesamiento de Lenguaje Natural (PLN)

3.3.1 Introducción al procesamiento de lenguaje natural

El Procesamiento del Lenguaje Natural (PLN) es un campo de estudio dentro de la Inteligencia Artificial (IA) que se enfoca en permitir que las computadoras comprendan, interpreten y generen lenguaje humano de manera efectiva [31]. Este campo desempeña un papel importante en la creación de sistemas inteligentes capaces de interactuar de manera natural con los usuarios y de comprender la información contenida en grandes volúmenes de texto.

Para el análisis del PLN, existe un aspecto de relevancia fundamental: el tratamiento de palabras, el cual se analizará a continuación.

3.3.2 Tratamiento de palabras en el PLN

En el procesamiento del lenguaje natural (PLN), el tratamiento de las palabras implica diversas técnicas y estrategias para pre procesar y manipular las palabras de manera que puedan ser utilizadas de manera efectiva en tareas como la traducción, clasificación de texto, extracción de información, entre otras.

Estas técnicas de tratamiento de palabras son esenciales en el procesamiento del lenguaje natural y se utilizan en diversas aplicaciones para mejorar la precisión y el rendimiento de los modelos de PLN. Al combinar estas técnicas de manera efectiva, es posible realizar análisis de texto más avanzados y obtener información útil a partir de grandes volúmenes de datos de texto.

De acuerdo con [32], las técnicas más comunes de tratamiento de palabras en el contexto del PLN son las siguientes:

1. **Tokenización:** Es el proceso de dividir el texto en unidades más pequeñas, llamadas tokens, que pueden ser palabras individuales, símbolos de puntuación o incluso caracteres. La tokenización es el primer paso en el procesamiento de texto y proporciona la base para análisis posteriores.
2. **Normalización:** Consiste en transformar las palabras a una forma estándar para reducir la variabilidad. Esto puede incluir la eliminación de mayúsculas, la lematización (reducción de las palabras a su forma base o lema) y la eliminación de caracteres especiales o dígitos.
3. **Stopwords:** Son palabras comunes que generalmente se eliminan del texto durante el preprocesamiento debido a que no aportan significado importante para el análisis. Ejemplos de stop words en español incluyen "el", "la", "de", "en", entre otros.
4. **Stemming:** Es el proceso de reducir las palabras a su raíz o base, eliminando prefijos y sufijos. A diferencia de la lematización, el stemming no garantiza que la palabra resultante sea una palabra real o válida en el idioma.
5. **Vectorización:** Consiste en representar las palabras como vectores numéricos en un espacio vectorial. Esto facilita el procesamiento posterior mediante algoritmos de aprendizaje automático, ya que los modelos generalmente requieren entradas numéricas.
6. **Embeddings:** Son representaciones vectoriales de palabras que capturan su significado semántico en un espacio multidimensional. Los embeddings se generan utilizando técnicas como Word2Vec, GloVe o FastText, y se utilizan ampliamente en tareas de PLN para capturar relaciones semánticas entre palabras.

En la siguiente sección, se detallará el concepto de PLN pipelines, los cuales son elementos esenciales para el procesamiento de un texto siguiendo distintas estrategias.

3.3.2 Tuberías de procesamiento en PLN

Una herramienta muy útil al tratar el lenguaje natural son las tuberías de procesamiento. Una tubería de procesamiento hace referencia a una serie de pasos que transforman datos de texto sin procesar en una salida deseada, como una etiqueta, un resumen o una respuesta [33].

Las tuberías de procesamiento o PLN pipelines son importantes porque ayudan a organizar, optimizar y simplificar los proyectos de PLN. Al seguir una pipeline clara y consistente, se garantiza que los datos estén limpios, estandarizados y listos para el análisis. También se puede reutilizar y modularizar el código, evitar la duplicación y los errores, mejorar el rendimiento y la escalabilidad. Además, se pueden tanto rastrear como evaluar los resultados, identificar, resolver problemas, así como actualizar y mejorar los modelos [34].

3.3.2.1 Diferencia de Tubería y Preprocesamiento

El preprocesamiento de datos implica todas las transformaciones y manipulaciones que se realizan en los datos antes de alimentarlos a un modelo de aprendizaje automático. Por otro lado, una tubería es una secuencia de pasos que se aplican secuencialmente a los datos, desde el preprocesamiento hasta la construcción del modelo y la evaluación del mismo. Por lo que podemos diferenciar que el procesamiento de datos es un componente clave dentro de una

tubería de aprendizaje automático. Mientras que el preprocesamiento se centra en la manipulación de los datos en sí, el pipeline abarca todo el flujo de trabajo de modelado de datos, desde la preparación inicial hasta la evaluación final del modelo [35].

3.3.3 Técnicas de generación de incrustaciones

De acuerdo con [36], las técnicas de generación de incrustaciones son métodos utilizados en el campo del procesamiento del lenguaje natural (PLN) para crear representaciones numéricas de palabras y textos que capturan la semántica y el contexto. Estas representaciones, conocidas como incrustaciones de palabras o *word embeddings*, se utilizan en diversas tareas de PLN, como análisis de sentimientos, clasificación de textos y generación de lenguaje.

Estos word embeddings pueden ser entrenados y utilizados para identificar similitudes y relaciones entre palabras. Al asignar a cada palabra un conjunto único de dígitos, digamos 100, 200 o incluso más, se logra una comprensión más profunda de la palabra. Por ejemplo, se pueden representar las palabras "madre" y "padre" con conjuntos distintos de dígitos, lo que permite entender mejor el contexto de cada palabra. Las representaciones vectoriales creadas mediante este proceso exhiben propiedades sorprendentes. Al restar el vector correspondiente a "Hombre" del vector de "Rey," el resultado es casi idéntico al resultado de restar "Mujer" del vector de "Reina." Incluso más asombroso, la resta de "Caminó" a "Caminando" se asemeja a la resta de "Nadó" a "Nadando." Estos ejemplos demuestran que el modelo no solo ha aprendido el significado y la semántica de estas palabras, sino también en cierta medida, la sintaxis y la gramática [36].

Esta propiedad del espacio de embeddings, donde palabras semánticamente similares se encuentran a una distancia cercana, es fundamental en el procesamiento del lenguaje natural. Al capturar estas relaciones semánticas, los modelos de word embeddings pueden proporcionar una comprensión más rica y contextualizada del lenguaje.

Uno de los modelos más destacados en este campo es Word2Vec, el cual es una técnica ampliamente utilizada para generar incrustaciones de palabras. Fue desarrollada por un equipo de investigadores de Google, liderado por Tomas Mikolov, y se introdujo por primera vez en 2013. Word2Vec es una red neuronal superficial que aprende a predecir la probabilidad de una palabra dada su contexto (CBOW) o el contexto dado una palabra (skip-gram). Word2Vec consta de dos arquitecturas principales: Continuous Bag of Words (CBOW) y Skip-Gram [36].

La arquitectura CBOW se centra en predecir la palabra central a partir de las palabras del contexto circundante. Las palabras del contexto se utilizan como entrada a la red neuronal, y la palabra central es la salida. Esta arquitectura comprende una capa de entrada, una capa oculta y una capa de salida. La capa de entrada representa las palabras de contexto codificadas, mientras que la capa de salida representa la palabra central codificada. La capa de incrustación es la que aprende la representación vectorial de las palabras, y la representación vectorial de la palabra central es la salida de la capa oculta [36] [37].

Por otro lado, la arquitectura skip-gram se enfoca en predecir las palabras del contexto circundante dada una palabra central. La palabra central es la entrada a la red neuronal, y las palabras de contexto son la salida. Al igual que en CBOW, esta arquitectura incluye una capa de

entrada, una capa oculta y una capa de salida. La representación vectorial de la palabra central es la salida de la capa oculta [36].

En general, las técnicas de generación de incrustaciones, como Word2Vec, se centran en aprender representaciones vectoriales de palabras que capturan relaciones semánticas y contextuales. Estas representaciones se utilizan para mejorar el rendimiento en diversas tareas de procesamiento del lenguaje natural al proporcionar una comprensión más profunda y eficaz del significado de las palabras en un contexto específico [38].

De acuerdo con lo establecido anteriormente, se explicaran los obstáculos específicos que enfrentan los sistemas de PLN en la creación de incrustaciones coherentes y significativas.

3.3.4 Desafíos en la generación de incrustaciones coherentes y relevantes

La generación de texto implica una serie de desafíos y consideraciones importantes, especialmente en el contexto de las representaciones de palabras, conocidas como word embeddings. Estas incrustaciones de palabras son esencialmente representaciones densas de palabras en forma de vectores de números reales. Así como una pintura puede ser una representación visual de una persona, un word embedding es una representación numérica de una palabra que codifica tanto su significado semántico como su contexto sintáctico en un formato comprensible por las computadoras [36], [37].

Sin embargo, uno de los desafíos más notables en el uso de word embeddings, como Word2Vec, es su incapacidad para manejar palabras desconocidas o fuera de vocabulario (OOV). Cuando el modelo se encuentra con una palabra que no ha sido previamente observada, carece de información para interpretarla y construir un vector adecuado para ella. En estas situaciones, se ve obligado a utilizar un vector aleatorio, lo cual no es ideal. Esto puede ser especialmente problemático en dominios como Twitter, donde los datos suelen ser ruidosos y dispersos, con palabras que pueden haber sido utilizadas solo una o dos veces en un corpus muy extenso. La resolución de este desafío es un área importante de investigación en el campo de word embeddings y generación de texto [36], [37].

La siguiente área fundamental es la tarea de texto a SQL (Text-to-SQL). En el siguiente capítulo, se presentará cómo los sistemas de procesamiento de lenguaje natural abordan el desafío de convertir consultas en lenguaje natural a consultas estructuradas en lenguaje SQL.

3.4 Introducción a la tarea text-to-SQL

3.4.1 Text to SQL

Text to SQL es una tarea del campo de la inteligencia artificial que busca la creación de modelos capaces de convertir preguntas en lenguaje natural a consultas estructuradas en SQL. Esta disciplina desempeña un papel crucial en la interacción entre humanos y sistemas de bases de datos, ya que permite a los usuarios formular consultas complejas de manera más intuitiva y natural [9].

La tarea de Text-to-SQL implica el desarrollo de algoritmos, modelos de procesamiento de lenguaje natural (PLN) y/o modelos basados en reglas [9], [39] que pueden comprender y analizar preguntas escritas en lenguaje natural y luego traducirlas de manera precisa a consultas SQL comprensibles por las bases de datos. Esta conversión requiere la capacidad de entender la semántica y la intención detrás de la pregunta, identificar entidades relevantes como atributos, tablas, condiciones y valores para estructurar la consulta de manera adecuada que permitan obtener la información solicitada [9].

La importancia de esta tarea radica en simplificar el acceso a bases de datos complejas, facilitando a los usuarios la extracción de información sin necesidad de tener un conocimiento profundo de SQL o de la estructura de la base de datos. Esto tiene aplicaciones significativas en ámbitos como la asistencia virtual, la búsqueda de información y el desarrollo de interfaces más amigables para acceder y gestionar datos de manera eficiente [9].

Continuando, ahora se presentarán las diferentes variantes de la tarea de texto a SQL.

3.4.2 Variantes de la tarea text to SQL

Aunque el objetivo principal de la tarea *text to SQL* es obtener una consulta de SQL a partir de un texto en lenguaje natural, se han definido algunas variantes de la tarea dependiendo de la dificultad que conlleva dicha conversión. Özcan et. al. identifican 4 categorías para las consultas con base en su dificultad [40]:

- Consultas con selecciones simples sobre una sola tabla
- Consultas con agregadores (MAX, MIN, COUNT, etc.) sobre una sola tabla que involucran agrupamiento (GROUP) y ordenamiento (ORDER BY)
- Consultas que involucran múltiples tablas (JOIN)
- Consultas que involucran consultas anidadas

Cada uno de estos niveles puede involucrar a uno o más de los niveles anteriores, lo cual da lugar a un gran espectro de posibles consultas, y por lo tanto, de casos por cubrir.

Después de explorar las distintas variantes de la tarea de texto a SQL, es importante considerar los conjuntos de datos disponibles para este propósito. En la siguiente sección, se analizarán los conjuntos de datos utilizados en la literatura sobre los sistemas de text-to-SQL, destacando sus características, alcance y relevancia para la tarea en cuestión.

3.4.3 Conjuntos de datos

En la actualidad, existen 2 grandes conjuntos de datos que cumplen con las características de tener una gran cantidad de consultas y un número decente de bases de datos en diversos contextos. Estos conjuntos se utilizan como punto de referencia para evaluar el desempeño de un modelo en esta tarea.

El primero de ellos es WikiSQL, el cual se enfoca en los dos primeros niveles de dificultad mencionados en la sección anterior (consultas simples con posible uso de agregadores, ordenamiento y/o agrupamiento) [17] y que además, por la forma en que fue diseñado, no es

necesario indicar el nombre de la tabla a la hora de redactar una posible consulta SQL, ya que sus bases de datos se conforman por una sola tabla.

Por otro lado, Spider es un conjunto de datos más completo, ya que incluye consultas de todos los niveles y, en comparación con WikiSQL, incluye bases de datos conformadas por múltiples tablas [17], lo cual abre el desafío de detectar la o las tablas necesarias para obtener la información que solicita el usuario.

Cabe mencionar que ambos conjuntos de datos tienen sus consultas redactadas en inglés, y a pesar de que existen proyectos con el objetivo de traducir conjuntos como Spider a idiomas distintos del inglés [19], a la fecha de redacción no han hecho públicos sus resultados.

Tras revisar los conjuntos de datos disponibles, en la siguiente sección se abordarán las técnicas de etiquetado y anotación de los datos utilizados en la tarea de texto a SQL.

3.4.5 Etiquetado y anotación de datos

La tarea de etiquetado es esencial en el reconocimiento de patrones, y consiste sencillamente en asignar una (o varias) etiquetas a cada dato en nuestro conjunto de datos, lo cual será de utilidad a la hora de entrenar un modelo de aprendizaje automático.

Los tipos de etiquetas que podemos asignar a un dato varían con respecto a la tarea que se desea resolver, pero en el caso de la clasificación, existen 3 principales [41]:

- Etiqueta binaria: Se utiliza cuando existen 2 clases, y solo una de ellas es posible a la vez. Puede verse con frecuencia en problemas donde las clases son “Sí” y “No”.
- Multi-clase: Se utiliza cuando existen más de 2 clases, pero solo una de ellas es posible a la vez.
- Multi-etiqueta: Se utiliza cuando existen 2 o más clases, con más de una posible a la vez.

Por otro lado, la anotación de datos consiste en el proceso de agregar anotaciones (o metadatos) a los datos originales, lo cual tiene como beneficio potencial un mejor desempeño a la hora de entrenar un modelo de aprendizaje automático [42]. Al igual que con las etiquetas, los tipos de anotaciones que se pueden realizar dependen del tipo de datos y de la tarea a resolver, pero para el desarrollo de este trabajo, nos centraremos en la anotación de texto, la cual consiste en asignarle una etiqueta a una o varias secciones del texto.

Tras analizar el etiquetado y la anotación de los datos, es crucial definir métricas de evaluación efectivas para medir el rendimiento de los modelos de text-to-SQL. A continuación, se presentan las métricas clave.

3.4.6 Métricas de evaluación, como Execution Accuracy y Logical Form Accuracy

La evaluación de modelos de generación de lenguaje natural para consultas SQL es una parte fundamental de la investigación en el campo de la inteligencia artificial y el procesamiento de lenguaje natural. Como se observó en [17] dos métricas de evaluación comunes utilizadas para

medir el rendimiento de estos modelos son *execution accuracy* y *logical form accuracy*. Estas métricas son esenciales para determinar la precisión y efectividad de los modelos en la generación de consultas SQL a partir de texto natural.

Execution accuracy:

- Se refiere a la capacidad del modelo para generar consultas SQL que, cuando se ejecutan en una base de datos, devuelve los resultados correctos. Esta métrica mide la precisión de las consultas generadas por el modelo en términos de su capacidad para recuperar información precisa de la base de datos. Un alto valor de "execution accuracy" indica que el modelo es capaz de generar consultas que funcionan correctamente y proporcionan resultados precisos. [17]
- Fórmula:

$$\text{Execution Accuracy} = (\text{Número de Consultas SQL Correctas}) / (\text{Número Total de Consultas en el Conjunto de Prueba})$$

Logical form accuracy:

- Se enfoca en la estructura y sintaxis de las consultas SQL generadas por el modelo. Mide qué tan precisamente el modelo puede generar la forma lógica de una consulta SQL, independientemente de si se ejecuta correctamente en la base de datos. Esta métrica evalúa la coherencia y la estructura de las consultas generadas, lo que es crucial para garantizar que sean interpretables y utilizables por los usuarios [17].
- Fórmula:

$$\text{Logical Form Accuracy} = (\text{Número de Consultas SQL con Forma Lógica Precisa}) / (\text{Número Total de Consultas en el Conjunto de Prueba})$$

Un ejemplo ilustrativo de estas métricas sería un modelo que recibe la siguiente consulta en lenguaje natural: "Encuentra todos los empleados con un salario superior a \$50,000". El modelo debe generar una consulta SQL que sea tanto precisamente ejecutable como tenga una forma lógica precisa. Una consulta SQL precisa podría ser:

"SELECT * FROM empleados WHERE salario > 50000;"

Si esta consulta se ejecuta correctamente en la base de datos y recupera la información correcta, el modelo obtendría una alta *execution accuracy*. Al mismo tiempo, si la consulta SQL generada sigue una estructura lógica adecuada y es interpretable, se consideraría que el modelo tiene una alta *logical form accuracy*.

Las métricas de evaluación como *execution accuracy* y *logical form accuracy* son esenciales para determinar el rendimiento de los modelos de generación de consultas SQL a partir de texto natural. Estas métricas garantizan que las consultas generadas sean precisas, ejecutables y tengan una estructura lógica adecuada, lo que es fundamental para aplicaciones prácticas en bases de datos y sistemas de procesamiento de lenguaje natural.

Por otro lado tenemos Top-N Accuracy como métrica de evaluación. Top-N Accuracy mide con qué frecuencia la clase predicha se encuentra dentro de los N valores principales de tu distribución softmax [43]. En palabras más simples, Top-N Accuracy considera un acierto si la clase correcta se encuentra entre las N clases con las probabilidades más altas predichas por el modelo.

Un ejemplo de top N accuracy, supongamos que tenemos un modelo de clasificación de imágenes de animales dónde tenemos 4 clases: Dog, Cat, Horse y Mule.

Image#	True label	Predicted Label	Top 2 predicted label
1	Dog	Dog	Dog, Cat
2	Dog	Cat	Cat, Horse
3	Cat	Cat	Cat, Dog
4	Horse	Mule	Mule, Horse
5	Mule	Horse	Horse, Dog

Fig 3.9. Ejemplo Top-N Accuracy. Extraído de [43].

Como se puede observar en la figura 3.8, la columna “Predicted Label” contiene las predicciones Top 1, mientras la columna “Top 2 predicted label” contiene las predicciones Top 2. En este caso para las predicciones Top 1, solo acertó en 2 de 5 ocasiones, por lo que el accuracy es de 40%, mientras que en el caso Top 2, acertó en 3 de 5 ocasiones por lo que su accuracy es de 60%.

Finalmente se utilizó la distancia de levenshtein como métrica de evaluación. La distancia de Levenshtein es una medida de similitud entre dos cadenas, que toma en cuenta el número de operaciones de inserción, eliminación y sustitución necesarias para transformar una cadena en la otra [44]. Se utiliza para comparar cambios requeridos en las consultas generadas por el modelo con las consultas reales del dataset.

3.5 Desafíos y Limitaciones

3.5.1 Ambigüedad en consultas en lenguaje natural

La principal dificultad en los procesos de recuperación de información utilizando lenguajes formales no radica en cuestiones técnicas, sino en aspectos psicológicos, como comprender la necesidad real del usuario y la correcta formulación de sus preguntas o requerimientos [45]. Una dirección muy prometedora para abordar este problema es la utilización del lenguaje natural. Sin embargo, uno de los desafíos más significativos en el Procesamiento del Lenguaje Natural surge cuando una expresión en lenguaje natural tiene múltiples interpretaciones, es decir, cuando puede asignarse más de un significado en el idioma de destino. Esta problemática de la ambigüedad se presenta en todos los niveles del lenguaje, sin excepción [39].

Supongase que un usuario solicita la siguiente consulta en lenguaje natural: "Encuentra los empleados que trabajan en el proyecto de software". Esta frase puede generar ambigüedad en cuanto a qué proyecto de software se refiere, ya que puede haber varios proyectos de software en una base de datos.

El desafío aquí es determinar a cuál proyecto de software se refiere el usuario. La máquina debe comprender el contexto y la relación entre las entidades (empleados y proyectos de software) para generar la consulta SQL adecuada. Esto puede involucrar preguntas adicionales al usuario o inferir la intención basándose en el contexto. Por ejemplo:

1. Si el usuario está trabajando en un contexto específico y ya ha hablado de un proyecto de software en particular, la máquina podría inferir que se refiere a ese proyecto.
2. Si no se dispone de información contextual, la máquina podría solicitar al usuario que especifique el proyecto de software por nombre u otros atributos distintivos.
3. En un escenario más avanzado, la máquina podría utilizar técnicas de procesamiento de lenguaje natural y aprendizaje automático para inferir la intención del usuario basándose en sus interacciones anteriores y en la información disponible en la base de datos.

La resolución de ambigüedades semánticas y la interpretación precisa de relaciones implícitas son desafíos cruciales en la traducción de lenguaje natural a consultas SQL, ya que afectan directamente la capacidad de la máquina para comprender y atender las solicitudes de los usuarios de manera efectiva [45].

El desafío central de este proyecto, reside en la complejidad de la ambigüedad y el análisis contextual, además de la variedad de expresiones que se pueden utilizar para hacer referencia a la retribución de los mismos datos, lo cual requiere de una gran cantidad (posiblemente infinita) de patrones registrados para realizar una predicción que vaya acorde con la intención del usuario.

CAPÍTULO 4. Análisis y diseño del modelo

4.1 Alcance

Realizar un proyecto como el que se propone enfrenta como principales desafíos el discernimiento entre diferentes interpretaciones posibles de una misma consulta en lenguaje natural, la consideración del contexto de la consulta (y con ello, el uso de lenguaje propio del contexto de la base de datos) y, finalmente, la falta de patrones para interpretar una consulta en lenguaje natural impidiendo la generación de una consulta SQL precisa. Resolver esta tarea, o cualquier otra que involucre lenguaje natural, requiere una comprensión profunda del idioma y de los diversos contextos que se pueden abordar, lo cual, incluso a día de hoy, no ha conseguido una solución definitiva. Por todos estos motivos, limitaremos el alcance de este proyecto como se indica a continuación:

El modelo tendrá como objetivo principal generar consultas SQL sintácticamente válidas, sin garantía de que sean semánticamente correctas, a partir del esquema de la base de datos que se desea consultar, y de una consulta en lenguaje natural (español), la cual deberá contener

suficiente información para generar su equivalente en SQL, así como cumplir con alguno de los patrones cubiertos por el modelo.

Adicionalmente, las consultas que podrá generar el modelo se limitarán a consultas del tipo *SELECT-FROM-WHERE*, donde la condición sólo podrá conformarse por un solo operador lógico, que a su vez deberá de pertenecer a alguno de la siguiente lista: <, >, =, <=, >=, !=.

Finalmente, en caso de que el modelo se encuentre frente a una situación en la cual sea incapaz de generar una respuesta válida, este será capaz de abstenerse.

4.2 Requerimientos funcionales del modelo

Los requerimientos funcionales son especificaciones detalladas y concretas que describen las funciones, servicios o acciones que un sistema, software o producto debe cumplir. En el desarrollo de software, estos requerimientos definen las capacidades y comportamientos específicos que el sistema debe tener para satisfacer las necesidades del usuario, el cliente o el negocio [46].

A continuación, listamos los requerimientos funcionales considerados para el desarrollo de este modelo:

- El modelo deberá ser capaz de generar, cuando le sea posible, consultas en lenguaje SQL a partir de una entrada conformada por una consulta en lenguaje natural (español) y el esquema de la base de datos que se desea consultar.
- El modelo deberá entregar consultas SQL válidas
- En caso de no poder generar una consulta SQL válida a partir de la información proporcionada, el modelo deberá tener la capacidad de abstenerse
- En caso de que el modelo obtenga más de una respuesta válida, sin la seguridad de que sea correcta, el sistema entregará todas las posibles respuestas
- El modelo deberá ser capaz de manejar consultas que involucren la selección de uno o más atributos.
- El modelo deberá ser capaz de interactuar con bases de datos que tengan más de una tabla.
- El modelo deberá ser capaz de inferir la tabla con la que se está interactuando sin necesidad de que esta sea mencionada explícitamente cuando esto le sea posible
- El modelo deberá ser capaz de inferir los atributos consultados sin necesidad de que sean escritos en la consulta en español exactamente como aparecen en el esquema de la base de datos (cuando esto sea posible)
- El modelo deberá ser capaz de detectar condiciones simples (conformadas por un solo operador lógico perteneciente a la siguiente lista: <, >, =, <=, >=, !=) cuando sea necesario (en caso de ser posible)

4.3 Arquitectura del modelo

La idea principal de ñ2SQL es que permita la generación de consultas SQL a partir de consultas hechas en lenguaje natural en el idioma español, por medio de un modelo basado en

reconocimiento de patrones y en aprendizaje automático. En esta sección, describiremos los módulos de Ñ2SQL y cómo son aplicados para resolver la tarea *Text to SQL*.

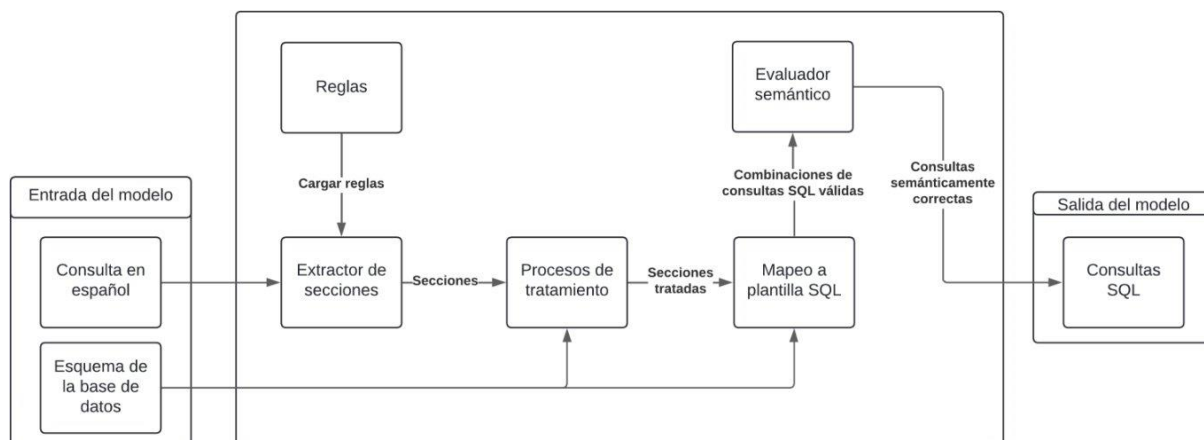


Fig 4.1. Diagrama de la arquitectura que proponemos, la cual llamamos ñ2SQL.

4.3.1 Identificador de secciones relevantes

Con base en los patrones identificados en la etapa de etiquetado y refinamiento de reglas, este bloque extraerá las secciones de texto de la consulta que considere relevantes para generar la consulta SQL (correspondientes a alguno de los bloques que conforman una consulta SQL básica, ya sea el nombre de la tabla, de los atributos, o la condición).

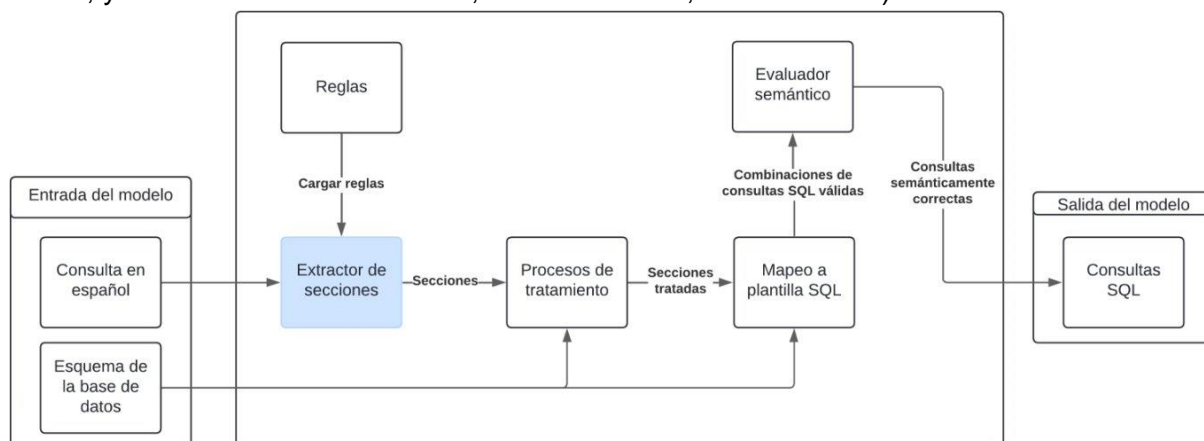


Fig 4.2. Diagrama de la arquitectura haciendo énfasis en la extracción de secciones.

4.3.2 Tratamiento de secciones extraídas

Con las secciones extraídas de la sección anterior, se utilizarán múltiples tuberías de tratamiento, de modo que se obtengan sólo los datos relevantes para la sección a tratar, ya sean nombres de tablas, atributos, operadores lógicos o atributos condicionales.

Como nuestra tubería base, y contra la que se comparan el resto de experimentos, proponemos el uso de una estrategia básica, la cual asumirá que las secciones extraídas no necesitan ningún tratamiento adicional, y usará el texto encontrado en dicha sección para generar las consultas SQL. En el caso de las condiciones, se hace uso de un tratamiento mínimo, en el cual se utilizan

dos conjuntos de reglas, uno para extraer los operadores lógicos de una sección, y otro para extraer los atributos y valores condicionales; y una vez extraídas estas secciones, se usa la estrategia básica nuevamente y se forma una condición potencial.

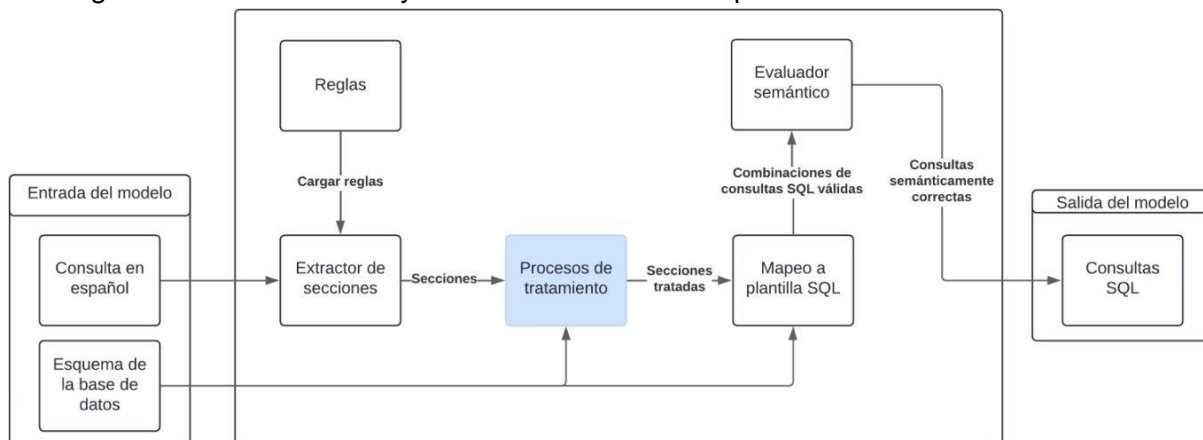


Fig 4.3. Diagrama de la arquitectura haciendo énfasis en el proceso de tratamiento.

4.3.3 Mapeo a plantilla SQL

Una vez obtenidos los datos correspondientes a cada sección de la consulta SQL, estos se mapean a una plantilla SQL para obtener el equivalente de la consulta en SQL. En esta etapa se generan todas las combinaciones válidas de consultas posibles con base en las secciones tratadas resultantes de la etapa anterior (sin darle importancia a si las consultas generadas son semánticamente correctas, ya que esto se abordará más adelante).

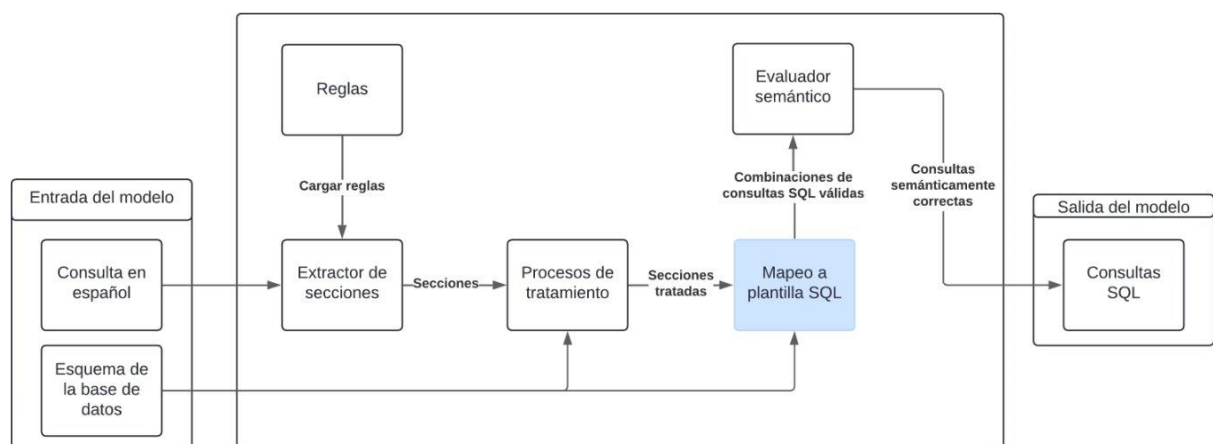


Fig 4.4. Diagrama de la arquitectura haciendo énfasis en el mapeo a la plantilla SQL.

4.3.4 Evaluador semántico

Con la información del esquema de la base de datos y una potencial consulta SQL, se determinará si dicha consulta es correcta semánticamente, para lo cual se realizarán distintas acciones y verificaciones con el fin de verificar si la consulta tiene sentido semántico bajo el esquema proporcionado. Por ejemplo, si se tiene una consulta “SELECT alumno FROM edificio”, pero el esquema no cuenta con una tabla “edificio”, dicha consulta sería semánticamente incorrecta.

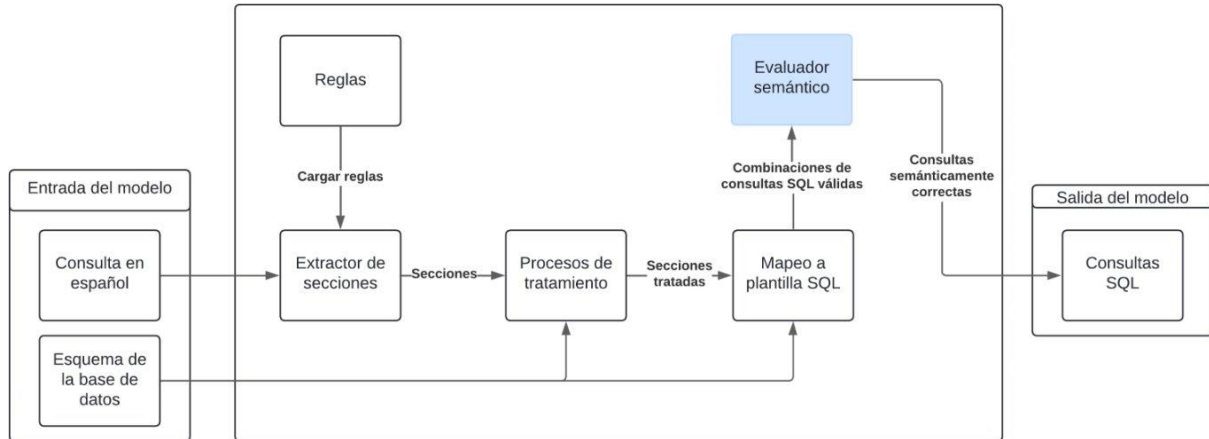


Fig 4.5. Diagrama de la arquitectura haciendo énfasis en el evaluador semántico.

4.4 Restricciones del modelo

En esta sección, presentamos las restricciones clave que podrían influir en la eficacia y exactitud del modelo. Estas restricciones abordan aspectos como la complejidad del procesamiento del lenguaje natural, las limitaciones en la generalización de patrones, la interpretación ambigua y contextual, así como las posibles limitaciones tecnológicas que podrían impactar en el rendimiento del modelo:

1. Complejidad de Interpretación Lingüística:

- La complejidad del procesamiento del lenguaje natural podría limitar la capacidad del modelo para interpretar de manera precisa ciertas estructuras lingüísticas más complejas o ambigüedades a algunos patrones del habla. Esto podría conducir a desafíos en la generación exacta de consultas SQL en todos los casos posibles.
- Las consultas que podrá generar el modelo se limitarán a consultas del tipo *SELECT-FROM-WHERE*, donde la condición sólo podrá conformarse por un solo operador lógico, que a su vez deberá de pertenecer a alguno de la siguiente lista: $<$, $>$, $=$, $<=$, $>=$, $!=$.

2. Limitaciones en la Generalización:

- La capacidad del modelo para generalizar y adaptarse a patrones y contextos no previstos en el conjunto de datos puede estar restringida por la cantidad y diversidad de ejemplos proporcionados en la fase de entrenamiento. Esto puede resultar en interpretaciones menos precisas en situaciones menos comunes o con patrones no explícitamente abordados en la fase de desarrollo.

3. Interpretación Ambigua y Contextual:

- La interpretación de consultas en lenguaje natural, que a menudo es subjetiva y contextual, puede generar ambigüedades difíciles de resolver. El modelo podría enfrentar dificultades en la comprensión precisa del contexto, lo que podría conducir a múltiples interpretaciones o a la imposibilidad de generar consultas SQL válidas en ciertas situaciones.
- 4. Ausencia de memoria:**
- El modelo no retiene ni utiliza información o consultas pasadas para influir en la generación actual de consultas SQL, por lo cual el mismo no tiene la capacidad de actuar como un agente conversacional capaz de mantener conversaciones o diálogos continuos.
- 5. Limitaciones Tecnológicas:**
- El rendimiento del modelo puede verse afectado por limitaciones tecnológicas, como la capacidad computacional o el tiempo requerido para procesar consultas más complejas. Estas limitaciones podrían influir en la rapidez y eficacia de la generación de consultas SQL.

4.5 Tecnologías a utilizar

El proyecto requerirá una combinación de tecnologías. A continuación, se detallan las tecnologías principales que se emplearán:

- 1. Python como Lenguaje de Programación:**
 - Se utilizará Python debido a su versatilidad y a la amplia gama de librerías disponibles para el procesamiento del lenguaje natural (NLP, por sus siglas en inglés). La facilidad de uso y la robustez de Python lo convierten en una opción ideal para implementar algoritmos de análisis lingüístico.
- 2. Librerías como spaCy:**
 - SpaCy es una librería NLP de código abierto en Python que ofrecerá las capacidades necesarias para realizar tareas cruciales en el procesamiento del lenguaje, como tokenización, análisis sintáctico y reconocimiento de entidades. Su eficacia y velocidad en el análisis lingüístico serán fundamentales para traducir las consultas de manera precisa.

La combinación de estas tecnologías proporcionará las herramientas necesarias para el desarrollo de una solución completa y efectiva.

4.6 Análisis de costos

En esta sección presentamos un análisis detallado de los costos involucrados en el desarrollo del proyecto durante un período estimado de 10 meses. El análisis contempla diversos aspectos relacionados con los recursos necesarios para la ejecución del trabajo terminal por parte de un equipo conformado por 4 estudiantes universitarios.

En la Tabla I detallamos los costos asociados a equipamiento, servicios, y otros gastos esenciales para llevar a cabo el proyecto:

Tabla I
Análisis detallado de los costos asociados en la elaboración del proyecto

Concepto	Descripción	Unidades por persona	Costo Unitario (promedio)	Costo Total (por persona)	Costo Total (equipo de 4 personas)
Equipamiento	Computadoras, software, herramientas de trabajo	1 equipo	\$15,000	\$15,000	\$60,000
Internet	Costo mensual de conexión a internet	10 meses	\$200	\$2,000	\$8,000
Espacio de trabajo	Alquiler o gastos asociados al lugar de trabajo	10 meses	\$300/mes	\$3,000	\$12,000
Capacitación	Cursos, libros u otros recursos educativos	2 cursos	\$200	\$400	\$1,600
Total					\$81,600

4.7 Análisis de riesgos

El análisis de riesgos es fundamental para identificar, evaluar y gestionar posibles obstáculos o contratiempos que podrían surgir durante la ejecución del trabajo terminal.

Hemos identificado y evaluado diversos riesgos potenciales, considerando la falta de experiencia, posibles cambios en los requerimientos, problemas de conectividad, limitaciones de recursos y otros factores que podrían influir en el progreso y la finalización exitosa del proyecto.

Cada riesgo ha sido evaluado en términos de su prioridad, indicando el impacto potencial en el proyecto, y su probabilidad de ocurrencia. Este análisis ayuda a establecer una base sólida para la planificación de estrategias de mitigación y gestión de riesgos, permitiendo al equipo anticipar y abordar posibles obstáculos de manera efectiva durante el desarrollo del trabajo terminal.

En la Tabla II, enlistamos cada uno de los riesgos identificados:

<p>Tabla II</p> <p>Análisis de riesgos asociados al desarrollo del proyecto</p>			
Concepto	Descripción	Prioridad (Alta, Media, Baja)	Probabilidad (Alta, Media, Baja)
Falta de experiencia	Equipo con poca experiencia en traducción NL-SQL	Alta	Alta
Cambios en requerimientos	Modificaciones frecuentes de requisitos	Media	Media
Problemas de conectividad	Fallas en la conexión a internet	Baja	Baja
Limitaciones de recursos	Equipamiento insuficiente para la tarea	Media	Media
Errores en la traducción	Problemas de precisión en la conversión NL-SQL	Media	Media
Retrasos en entregas	Dificultades en el cumplimiento de plazos	Media	Media
Dependencia de terceros	Riesgo asociado a proveedores externos	Baja	Baja
Problemas de compatibilidad	Incompatibilidad de herramientas o sistemas existentes	Baja	Baja
Cambios en el equipo	Salida de miembros del equipo o cambios inesperados	Media	Media
Inestabilidad técnica	Problemas con la estabilidad o rendimiento de herramientas	Media	Alta

CAPÍTULO 5. Implementación del modelo

5.1 Recopilación de datos

5.1.1 Evaluación y selección de datasets

La evaluación y selección de datasets es un paso crucial en cualquier proyecto de investigación o desarrollo que involucra el procesamiento de datos. En este caso, se consideró fundamental elegir un dataset que ofreciera una variedad representativa de preguntas y consultas en lenguaje natural para SQL.

Se llevó a cabo una exhaustiva evaluación de varios datasets disponibles públicamente, centrándonos principalmente en Spider y WikiSQL debido a su relevancia en el ámbito de la generación de consultas SQL a partir de lenguaje natural.

Proceso de Evaluación:

1. **Recopilación de Información:** Se recopiló información detallada sobre ambos datasets, incluyendo su tamaño, complejidad, diversidad de preguntas, dominios cubiertos y calidad de los datos.
2. **Análisis de la Estructura del Dataset:** Se examinó la estructura de cada dataset para comprender cómo estaban organizadas las preguntas en lenguaje natural y las consultas SQL correspondientes. Se evaluó la coherencia y la claridad de la estructura de los datos.
3. **Exploración de la Complejidad de las Consultas:** Se analizó la complejidad de las consultas en cada dataset, considerando el número de tablas involucradas, la presencia de cláusulas como JOIN y GROUP BY, y la variedad de operaciones SQL.
4. **Revisión de la Diversidad de Preguntas:** Se revisó la diversidad de preguntas en lenguaje natural en cada dataset, prestando especial atención a la variedad de temas y dominios cubiertos. Se buscó un equilibrio entre preguntas simples y complejas.
5. **Exploración de la Calidad de los Datos:** Se evaluó la calidad de los datos en términos de precisión y consistencia. Se buscaron posibles errores o anomalías en los datos que pudieran afectar su utilidad para el proyecto.

Después de realizar la evaluación detallada de ambos datasets, se llevó a cabo un análisis comparativo para determinar cuál sería la mejor opción para el proyecto.

Análisis Comparativo:

Spider: Se destacó por su diversidad y complejidad de las preguntas en lenguaje natural, así como por la representación de consultas en un entorno más realista. Aunque tenía menos datos en comparación con WikiSQL, ofrecía una variedad más amplia de consultas y cubría una gama más amplia de temas.

WikiSQL: Si bien WikiSQL contenía una cantidad considerable de datos, las preguntas tendían a ser más simples y menos diversas en comparación con Spider. Además, la representación de las consultas SQL era más estándar y menos variada en términos de operaciones y cláusulas utilizadas.

Y después de una cuidadosa evaluación y análisis comparativo, se decidió seleccionar Spider como el dataset principal para el proyecto. Su diversidad y complejidad de preguntas en lenguaje natural proporcionaron una base sólida para el desarrollo y la evaluación de modelos de generación de consultas SQL.

5.1.2 Recopilar el dataset

Una vez seleccionado Spider como el dataset principal, se procedió a recopilar y preparar para su posterior procesamiento y análisis. La recopilación del dataset implicó la descarga de los archivos pertinentes y la organización de los datos en un formato adecuado para su uso en el proyecto.

Se llevaron a cabo pasos adicionales de limpieza y preprocesamiento para garantizar la integridad y la coherencia de los datos recopilados. Esto incluyó la eliminación de datos duplicados, la corrección de posibles errores y la estandarización de la estructura del dataset para facilitar su manipulación y análisis.

5.1.3 Análisis profundo de Spider

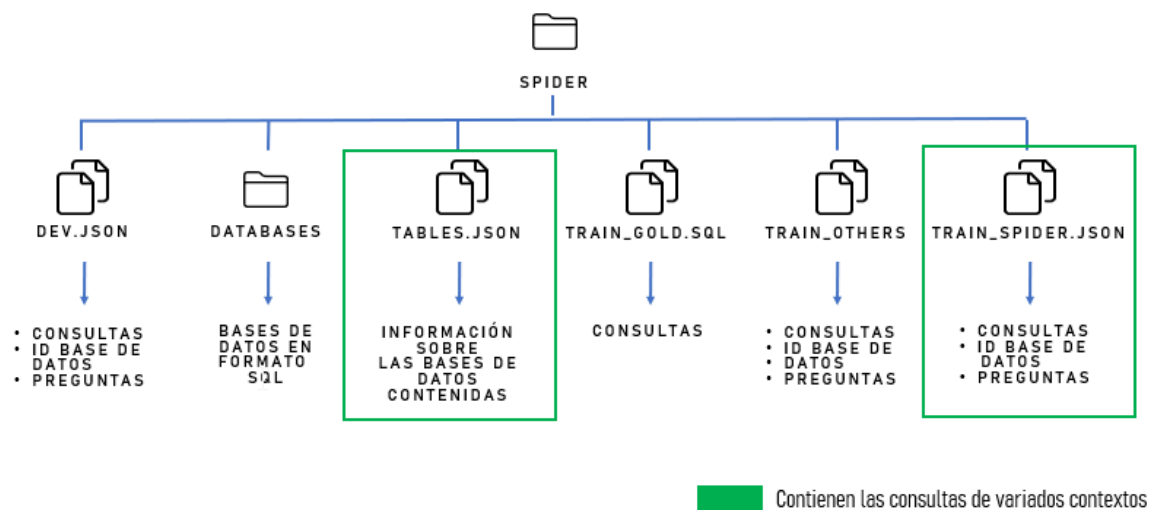


Fig 5.1. Estructura dataset Spider.

Una vez que el dataset Spider fue recopilado y preparado, se llevó a cabo un análisis profundo para comprender mejor su estructura, contenido y características clave. Este análisis incluyó la exploración de la distribución de las consultas en lenguaje natural y sus correspondientes consultas SQL, además de los esquemas de bases de datos que contiene.

Se examinaron diferentes aspectos del dataset, como la complejidad de las consultas, la diversidad de las preguntas y la cobertura de diferentes temas y dominios. Este análisis proporcionó información valiosa que ayudó a orientar el desarrollo y la implementación de modelos de generación de consultas SQL a partir de lenguaje natural, así como a establecer métricas de evaluación adecuadas para medir el rendimiento del modelo.

5.1.4 Preprocesamiento del Spider para su posterior tratamiento de traducción

Clave	Descripción
<code>db_id</code>	Identificador de la tabla
<code>query</code>	Consulta en SQL
<code>query_toks</code>	Tokens por los que está conformada la consulta SQL
<code>query_toks_no_value</code>	Valores de <code>query_toks</code> sin incluir valores específicos, solo indicando dónde se encuentra un valor
<code>question</code>	Pregunta en lenguaje natural equivalente a la consulta en SQL
<code>question_toks</code>	Tokens por los que está conformada la pregunta en lenguaje natural
<code>sql</code>	Información relacionada a la consulta SQL
<code>groupBy</code>	Booleano que indica si se utiliza el modificador GROUP BY en la consulta SQL
<code>having</code>	Booleano que indica si se utiliza el modificador HAVING en la consulta SQL
<code>orderBy</code>	Booleano que indica si se utiliza el modificador ORDER BY en la consulta SQL
<code>limit</code>	Booleano que indica si se utiliza el modificador LIMIT en la consulta SQL
<code>intersect</code>	Booleano que indica si se utiliza el modificador INTERSECT en la consulta SQL
<code>union</code>	Booleano que indica si se utiliza el modificador UNION en la consulta SQL
<code>except</code>	Booleano que indica si se utiliza el modificador EXCEPT en la consulta SQL

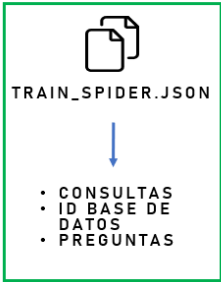


Fig 5.2. Estructura TRAIN_SPIDER.json.

Posterior al análisis del dataset Spider se tomaron únicamente 2 archivos que nos ayudarán con nuestra tarea. Estos archivos llevan por nombres “TABLES.JSON” y “TRAIN_SPIDER.JSON”, este último se puede observar en la Figura 5.2. El primero de ellos contiene toda la información sobre las bases de datos contenidas en el dataset, el segundo archivo contiene las preguntas en lenguaje natural, su equivalente en consulta sql y el ID de la base de datos a la que hace referencia.

El contenido de los archivos es demasiado grande, lo cuál afectará el proceso de traducción. Para solucionar este inconveniente solamente se hizo la traducción de ciertas secciones de ambos archivos. Los valores que se consideran esenciales para realizar el proceso de traducción son los nombres de las tablas, nombre de columnas y las consultas en lenguaje natural. Finalmente se realizaba un proceso de selección de estos valores con ayuda de las llaves en los archivos json.

5.1.5 Traducción automática

El proceso de traducción automática se realiza con ayuda de una librería de google translate y las secciones extraídas en el preprocesamiento. Específicamente utilizamos la librería *googletrans* que a su vez hace uso de Google Translate Ajax API.

La traducción se realizó con un script de python que recorre todos los valores que se querían traducir, sin embargo tuvimos algunas dificultades. Debido a la cantidad de datos que se deben

traducir tuvimos problemas con los límites de la librería, esta tenía un límite de traducción de palabras por segundo. Para solucionar este tema se añadieron intervalos de tiempo entre cada traducción para que la librería no detuviera el proceso y dejará la el proceso de traducción incompleto.

5.1.5.1 Corrección de traducciones incorrectas

Al realizar una revisión de las traducciones obtenidas se observa que algunas de estas carecen de sentido. Los problemas en las traducciones derivan de los textos originales, la librería elegida para realizar las traducciones utiliza los signos de puntuación, en este caso signos de interrogación de cierre, para delimitar las preguntas que serán traducidas. El problema entonces viene de la falta de signos de puntuación en algunas de las frases que se quieren traducir.

Para solucionar este problema se modifican los datos originales. Se identifican las partes dónde existe un salto de línea sin un signo de interrogación de cierre, en ese punto se agrega el signo faltante y finalmente con los nuevos datos se vuelve a realizar la traducción. Finalmente se logra identificar que las traducciones son correctas en su totalidad.

5.2 Anotación de las consultas en lenguaje natural

El enfoque que se propone requiere que se identifiquen las posibles secciones relevantes de la consulta en español, esto con el fin de reducir el espacio de búsqueda de los valores que formarán la consulta SQL, así que, partiendo de las consultas traducidas en el punto anterior, se anotan las mismas siguiendo dos tipos de etiquetas: de sección y de valor.

a)

CONDICION 1 ATRIBUTO 2 TABLA 3

¿Cuál es toda la información del cliente para los clientes en el estado de Nueva York?

b)

ATRIBUTO 1 VALOR 2 ATR_CONDICION 3 = 4 > 5 < 6 >= 7 <= 8

!= 9 AND 0 OR q

Encuentre los nombres oficiales de ciudades con una población mayor a 1500 o menor a 500.

Fig 5.3. Etiquetados realizados **a)** Ejemplo de etiquetas de sección. **b)** Ejemplo de etiquetas de valor.

Hacer uso de un enfoque donde las etiquetas que se pueden asignar a una anotación se definen de forma manual, resulta clave en el funcionamiento y extensión de nuestra arquitectura, ya que

permite definir nuevas etiquetas que resulten en el manejo de consultas de mayor complejidad sin necesidad de realizar grandes cambios en la arquitectura.

5.2.1 Etiquetas de sección

Como se mencionaba en el punto anterior, las etiquetas de sección tienen como objetivo principal responder a la pregunta “¿Dónde?” a la hora de realizar una búsqueda en la consulta en español de los elementos que conforman una consulta SQL. Saber que secciones de texto en específico potencialmente contienen información relevante para formar una consulta SQL tiene dos ventajas principales: la primera de ellas es la reducción de la zona de búsqueda de los valores, y la segunda es que, gracias a las etiquetas que refieren a alguna de las 3 secciones de una consulta SQL (atributos, nombre de la tabla, condición), es posible reducir aún más la búsqueda, de modo que se busquen únicamente aquellos elementos que coincidan con la etiqueta de la sección en la que se está analizando.

Para propósitos de resolver las consultas SQL de nivel 1, definiremos 3 etiquetas de sección, las cuales utilizaremos para anotar una parte de las consultas en español que conforman el conjunto de datos traducido:

1. Atributo: Sección de texto donde se mencionan los atributos a obtener en la consulta
2. Tabla: Sección de texto donde se hace referencia al nombre de la tabla que contiene la información a extraer.
3. Condición: Sección de texto con información para filtrar los datos de acuerdo con una condición lógica.

5.2.2 Etiquetas por valor

El propósito de estas etiquetas es señalar los valores específicos que se introducen en la consulta SQL para extraer la información solicitada. La diferencia entre las etiquetas de valor y las etiquetas de sección, es que, mientras que las etiquetas de sección intentan delimitar un área de búsqueda, las etiquetas de valor definen el valor que estamos buscando, ya sea el nombre de un atributo, de una tabla, entre otros.

Para propósitos de resolver las consultas SQL de nivel 1, se definen las siguientes etiquetas:

1. Atributo: Nombre del atributo a extraer de una tabla en cuestión.
2. Atributo condicional: Nombre del atributo que forma parte de una condición.
3. Valor condicional: Valor que deben tomar los datos pertenecientes a un atributo condicional.

Y, para la (o las) palabra(s) clave que indican el uso de un operador condicional, se definen las siguientes etiquetas, las cuales corresponden al operador lógico que se indica:

4. Igualdad
5. Menor que
6. Mayor que
7. Menor o igual que
8. Mayor o igual que
9. Distinto de
10. AND

5.3 Desarrollo del Modelo

5.3.1 Selección de algoritmos de aprendizaje automático

La elección del algoritmo adecuado desempeña un papel crucial en la precisión y eficiencia del proceso. Mientras que los algoritmos de aprendizaje automático han ganado popularidad en diversos campos, los algoritmos basados en reglas, como el algoritmo de reglas AQ, emergen como una alternativa sólida y transparente para la tarea de traducción de español a SQL.

Los sistemas basados en reglas se sustentan en una base de conocimiento que alberga información específica del dominio, restricciones y reglas temáticas. Con un motor de inferencia a la vanguardia, estos sistemas toman decisiones, ofrecen respuestas y generan conclusiones de manera precisa y predecible. Esta estructura, delineada por Moya-Rodríguez [47] utiliza, proporciona una sólida base para la traducción de sentencias SQL al español.

El algoritmo de reglas AQ seleccionado cumple con la salida en forma de reglas, lo cual es crucial para que sea compatible con nuestro modelo. Esta compatibilidad asegura que cada regla definida explícitamente se traduzca de manera precisa y comprensible, facilitando la interpretación [29].

Además, estos algoritmos ofrecen control y flexibilidad. Es posible ajustar las reglas según sea necesario para adaptarse a cambios en el dominio o requisitos del sistema. Esta capacidad de adaptación permite que las traducciones sean precisas y actualizadas.

5.3.1.2 Resultados de nuestra implementación del algoritmo Quasi-Optimal (AQ) Learning para la obtención de reglas

Para comenzar, se obtuvo un conjunto de datos, el cual consiste en consultas extraídas del dataset de Spider, donde fueron etiquetadas por elementos específicos, como la sección izquierda y derecha. Estos datos se organizaron en un archivo de Excel, con dos columnas representando las secciones y una tercera columna para la clase, permitiendo la evaluación de las reglas. Luego, se dividió en un ochenta por ciento para entrenamiento y un veinte por ciento para validación. Después del entrenamiento, se evalúan las reglas.

Una vez realizado el entrenamiento, uno de los resultados es el siguiente:

```
Model for class 'ATRIBUTO':
  Complex: {<*,de> v <se,los> v <sus,*> v <los,que>}


Model for class 'CONDICION':
  Complex: {<cuyo,end> v <escuelas,*> v <de,los> v <que,*> v <con,*> v <para,?> v <*,especialidades?> v <una,*> v <de,.> v
<un,*> v <la,end> v <cuyas,*> v <estudiantes,*>}

Model for class 'TABLA':
  Complex: {<del,con> v <de,?> v <las,con> v <*,ubicadas> v <de,con> v <los,end> v <los,que> v <al,*> v <las,en> v <*,music
ales.> v <el,cuyo> v <*,aprobados> v <*,musicales?> v <destino,*> v <de,cuyo> v <*,enumeradas?>}
```

Fig 5.4. Resultados del entrenamiento de nuestra implementación del algoritmo AQ para obtener los contextos de las secciones principales

En la figura 5.4, podemos observar que la generación de las reglas de salida están compuestas por una serie de conjunciones unidas por disyunciones, lo cual nos indica cómo pueden estar delimitadas cada una de las 3 secciones generales de una consulta en lenguaje natural.

Para la clase ATRIBUTO, podemos encontrar que la regla que cubrirá esta sección en lenguaje natural será:

A dark rectangular box containing the text: {<*,de> v <se,los> v <sus,*> v <los,que>}.

```
{<*,de> v <se,los> v <sus,*> v <los,que>}
```

Fig 5.5. Reglas para la clase ATRIBUTO

Esto significa que de encontrar una parte de la oración de entrada con alguno de estos contextos a la izquierda y a la derecha será clasificado como ATRIBUTO y posteriormente se le dará un tratamiento particular para extraer la sección clave de la oración de entrada. Este proceso es el mismo para las otras dos reglas de salida que vemos en la Figura 5.4. En la sección 5.3.3, se explica con más detalle los procesos posteriores al resultado de este entrenamiento.

Es así que las reglas generadas son de utilidad porque nos permite descomponerlas en reglas más pequeñas y poder realizar la generación de las consultas SQL; en los siguientes capítulos se presenta como es este proceso.

5.3.2 Diseño y creación de tubería de datos

La tubería de datos es una estructura fundamental para gestionar la transformación de datos de texto, en este caso las consultas de lenguaje natural, a un formato utilizable para su posterior procesamiento. En el proyecto, hemos implementado tres clases principales que componen la tubería de datos: TextPipeline y sus subclases SimplePipeline y EmbeddingPipeline.

5.3.2.1 TextPipeline

La clase abstracta TextPipeline proporciona un esquema base para realizar transformaciones de texto. Esta clase está diseñada para manejar la transformación de secciones de texto, donde cada sección puede representar una tabla, un atributo o una condición en una consulta SQL. La tubería de texto consta de dos componentes principales: un evaluador semántico y un extractor de secciones.

- **Métodos:**

- transform_sections: Realiza la transformación de secciones de texto sucio a secciones limpias.
- transform_section: Realiza la transformación de una sección de texto a una sección limpia o una condición.

5.3.2.2 SimplePipeline

- SimplePipeline es una clase que realiza la evaluación semántica basada en reglas fijas. Se compone de métodos para transformar secciones de texto y evaluar las diferentes clasificaciones de secciones, como TABLA, ATRIBUTO y CONDICIÓN. Cada método de evaluación verifica la existencia de la tabla, columna o condición en la base de datos y devuelve la sección si se encuentra.
- **Métodos:**
 - `extract_condition`: Realiza la extracción de los diferentes elementos de la condición: operador, valor condicional y atributo condicional.

Los operadores se extraen en el objeto `operators extractor` donde se utiliza el método `extract_exact_match`, que busca las coincidencias exactas de los operadores contenidos en el texto de `section`.

El atributo condicional y valor condicional se encuentran en el objeto `value extractor`, que recopila secciones de texto con el método `extract`, la extracción es de ambos elementos, por lo que se implementa una sección de filtrado en el que se van a diferenciar mediante sus clasificaciones de la sección, por lo que con una etiqueta se filtran las secciones si corresponden a un atributo condicional o valor condicional.

Una vez extraídos todos los elementos se devuelven al operador, atributo condicional y valor condicional siempre y cuando existan, sino se omite.

- `transform_section`: Realiza la transformación de una sección de texto a una sección limpia o una condición.

5.3.2.3 EmbeddingPipeline

`EmbeddingPipeline` es una herramienta crucial diseñada para llevar a cabo la evaluación semántica mediante el uso de incrustaciones de palabras. Esta clase, que funciona con un umbral de similitud, permite establecer la correspondencia entre los nombres mapeados en la base de datos y aquellos identificados por la tubería de incrustaciones, lo que brinda flexibilidad al ajustar la sensibilidad de la evaluación para adaptarse a distintos contextos y requisitos de precisión.

Para esta tarea en particular, hemos seleccionado utilizar la biblioteca de procesamiento de lenguaje natural en Python conocida como `Spacy`. Optamos por emplear el modelo pre-entrenado `es_core_news_md` de `Spacy` debido a su equilibrio óptimo entre precisión y eficiencia computacional para nuestro proyecto.

Las incrustaciones de palabras en el modelo `es_core_news_md` de `Spacy` se obtuvieron a partir de un entrenamiento con el algoritmo `FastText`, al cual se le proporcionaron extensos conjuntos de datos de texto, tales como `OSCAR Common Crawl` y `Wikipedia`. Este proceso de entrenamiento permite capturar el significado y la relación semántica entre las palabras en un espacio vectorial de alta dimensionalidad, estableciendo así una

base sólida para realizar comparaciones entre distintas palabras, abordando de esta forma el problema de la sinonimia.

Debido a la estructuración de la clase TextPipeline, las tuberías Simple y Embedding no están acotadas a ser las únicas tuberías que se pueden implementar. La clase TextPipeline está planeada para servir como una guía para futuras creaciones de otras tuberías, con diferentes enfoques, que puedan manejar la limpieza de secciones.

Es importante destacar que para obtener la similitud entre los nombres mapeados en la base de datos y aquellos identificados por la tubería de incrustaciones se utiliza la similitud coseno. La similitud coseno es una medida que se utiliza para calcular la similitud entre dos o más vectores. La similitud del coseno es el coseno del ángulo entre vectores. Los vectores suelen ser distintos a cero y están dentro de un espacio de producto escalar [48].

5.3.3 Manipulación post-entrenamiento de los resultados obtenidos por nuestra implementación del AQ

5.3.1 Extracción de palabras delimitadoras de secciones (reglas)

La obtención de reglas se realiza con ayuda de las palabras etiquetadas y su clasificación asociada. De cada palabra etiquetada se obtienen las palabras o signos inmediatos tanto a su izquierda como a su derecha, y estas palabras o signos son guardados como contexto a la izquierda y contexto a la derecha.

Para obtener una lista de palabras delimitadoras se considerarán solamente ciertos elementos. En primer lugar se obtienen solamente los contextos a la izquierda como a la derecha de cada palabra etiquetada y la clasificación de dicha palabra. Es importante resaltar que la palabra etiquetada en ningún momento se guarda en la lista, de esta manera nuestra lista está conformada únicamente por `ctx_i` (contexto a la izquierda de la palabra), `ctx_d` (contexto a la derecha de la palabra) y la clasificación de cada palabra.

El no guardar la palabra sobre la cuál se obtuvieron los datos tiene un objetivo. Se realizó con el fin de poder obtener datos con los que se puedan encontrar patrones en las palabras que rodean a las secciones (tablas, columnas, condiciones y atributos) en una consulta de lenguaje natural sobre una base de datos.

5.3.2 Generación de la consulta

La generación de consultas SQL es una parte fundamental del proyecto. Esta funcionalidad es manejada por la clase QueryGenerator, la cual se encarga de producir consultas semánticamente correctas basadas en una consulta de lenguaje natural y la información contenida en la base de datos especificada. A continuación se detallan los pasos fundamentales de la generación de la consultas:

5.3.2.1 Extracción de secciones relevantes de la consulta en lenguaje natural

Después del proceso de extracción de palabras delimitadoras, las secciones relevantes de la oración de entrada son recabadas. Se lleva a cabo utilizando un conjunto de reglas predefinidas en el paso anterior. Estas reglas están representadas por instancias de la clase Rule, las cuales especifican el contexto izquierdo (left_context), el contexto derecho (right_context) y la clasificación (classification) asociada.

Funcionamiento del proceso:

1. **Inicialización de las reglas:** Se inicializan las reglas predefinidas que se utilizarán para la extracción de secciones. Cada regla define el contexto izquierdo y derecho que se debe encontrar en la consulta para identificar una sección relevante, así como su clasificación asociada.
2. **Iteración sobre las reglas:** Para cada regla predefinida, se itera sobre la consulta de entrada en busca de coincidencias con el contexto especificado por la regla. Cuando se encuentra una coincidencia, se identifica la sección correspondiente y se crea un objeto de tipo Section.

5.3.3.2 Limpieza de Secciones con Pipeline

Luego del paso anterior estas son limpiadas y procesadas utilizando el pipeline de transformación de texto correspondiente. Esto garantiza que las secciones estén en un formato adecuado para su procesamiento siguiente.

Dependiendo del tipo de pipeline seleccionado, SimplePipeline o EmbeddingPipeline, se realizará la transformación de las secciones introducidas usando distintas estrategias, las cuales se definen en la sección 5.3.2.

5.3.3.3 Generación de posibles tripletas de elementos de una consulta SQL

Una vez que las secciones han sido limpiadas, el QueryGenerator procede a generar todas las combinaciones posibles de elementos de consulta, tales como tablas, atributos y condiciones. Estas combinaciones se utilizan como base para construir las consultas SQL.

La clase QueryGenerator implementa métodos para generar consultas a partir de tripletas de secciones, donde cada triplet representa una combinación válida de secciones extraídas de la consulta en lenguaje natural. Estas consultas son entonces evaluadas semánticamente para determinar su corrección.

5.3.3.4 Validación semántica de las consultas SQL creadas

Finalmente, las consultas SQL generadas son devueltas como una lista de objetos Query, y con la ayuda de la clase SemanticEvaluator cada elemento es verificado si es una consulta válida y semánticamente correcta que puede ser ejecutada sobre la base de datos para obtener los resultados deseados. Este proceso asegura que las consultas generadas sean precisas y útiles para los usuarios finales.

5.3.4 Descripción general de la transformación de una consulta en lenguaje natural a una sentencia SQL con un ejemplo:

1. Identificar el esquema de la base de datos que se utilizara en el formato esperado y la consulta en lenguaje natural:

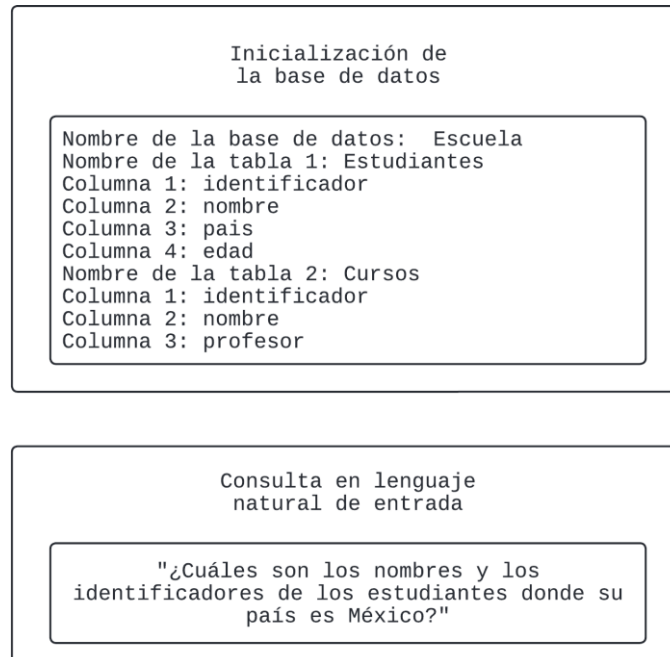


Fig 5.6. Paso 1: Entradas

2. Instanciar las reglas que se utilizarán para identificar las secciones relevantes de la consulta en lenguaje natural. Estas reglas son resultado del entrenamiento obtenido por el algoritmo AQ; sin embargo, para ese pequeño ejemplo solo son necesarias las siguientes reglas:

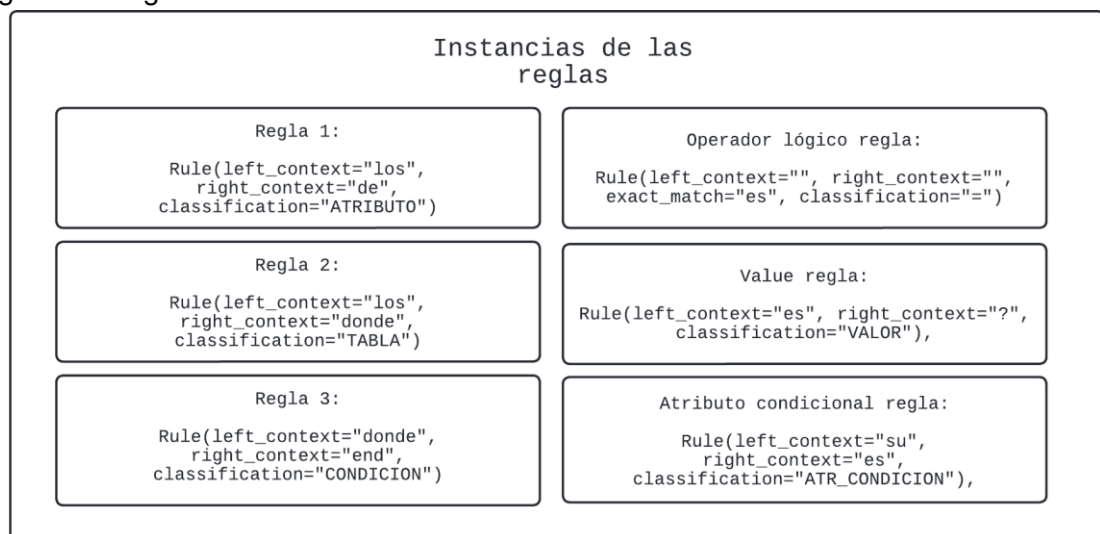


Fig 5.7. Paso 2: Instanciación de las reglas a utilizar

3. Definición de las tuberías de datos y otros parámetros de configuración necesarios para cada enfoque

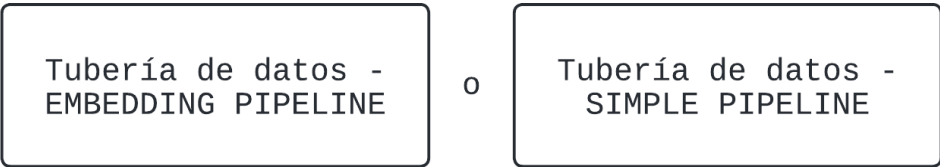


Fig 5.8. Paso 3: Configuración de las tuberías de datos

4. Proceder a la generación de la consulta:
- 4.1. A continuación, se extraen las secciones relevantes

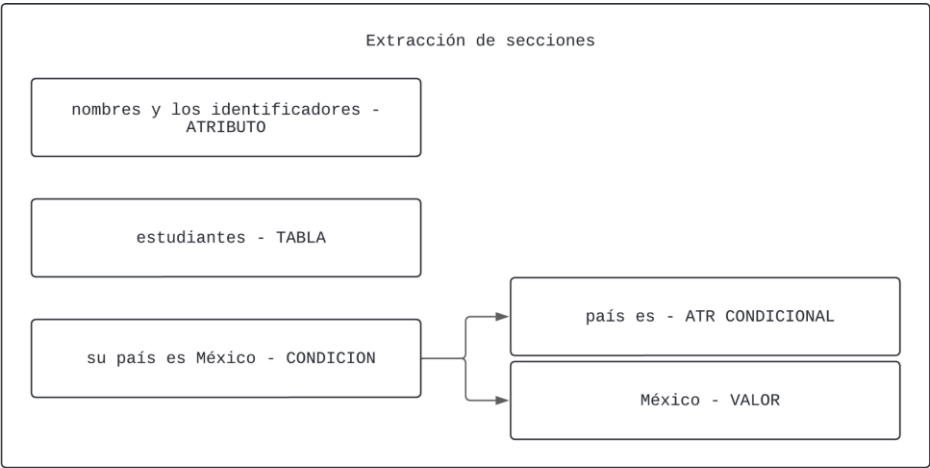


Fig 5.9. Paso 4: Extracción de secciones

- 4.2. Limpiar con la tubería de datos definida. El resultado dependerá completamente de la tubería escogida y su correspondiente enfoque. En este ejemplo, se utilizan los resultados que nos arrojaría la tubería EMBEDDING - PIPELINE.

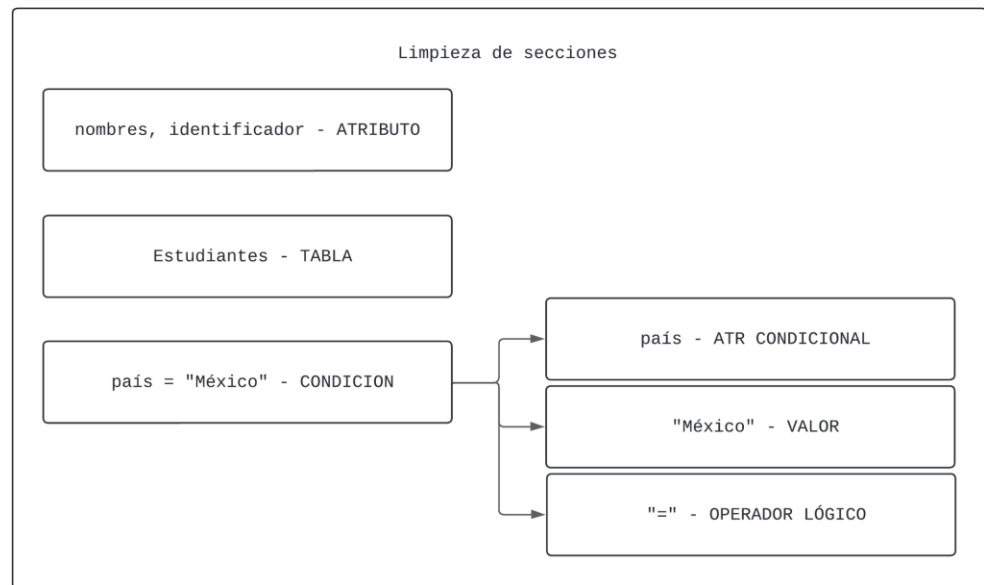


Fig 5.10. Paso 5: Limpieza de secciones

4.3. Generar las posibles tripletas válidas

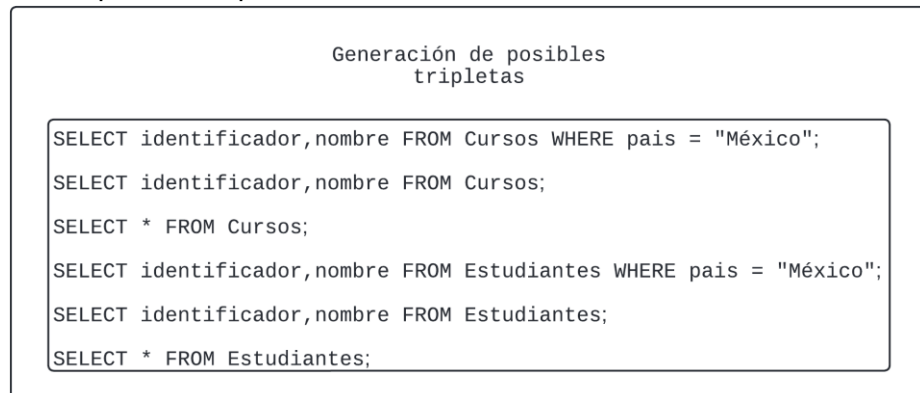


Fig 5.11. Paso 6: Generación de posibles tripletas

4.4. Validar aquellas que son semánticamente correctas

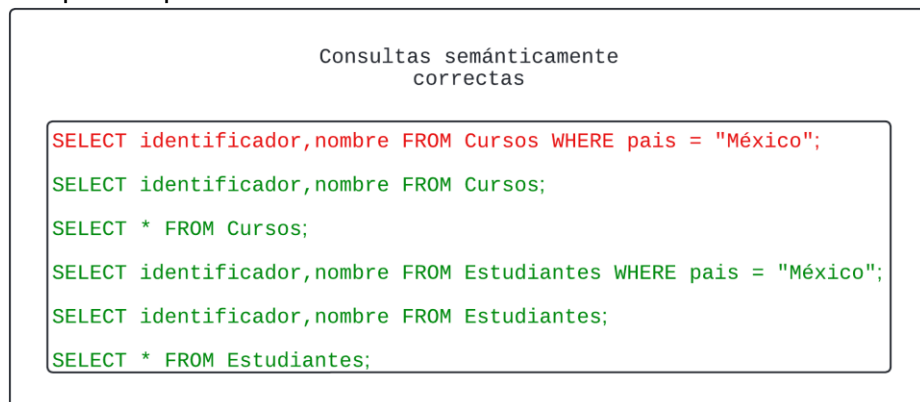


Fig 5.12. Paso 7: Validación de consultas a través del evaluador semántico

4.5. Obtener las consultas finales

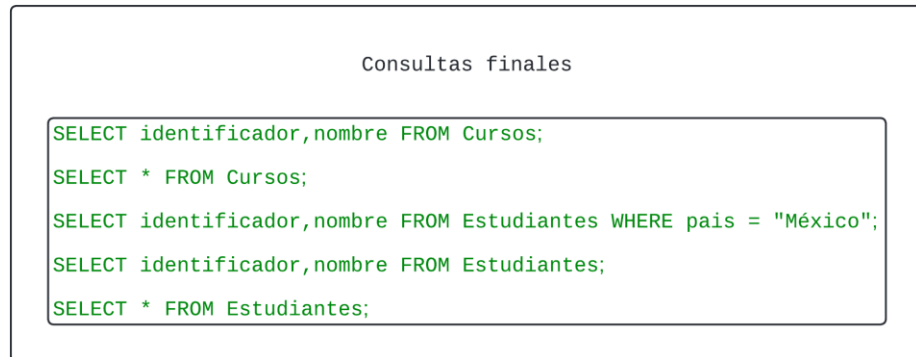


Fig 5.13. Paso 8: Obtención de consultas finales

CAPÍTULO 6. Pruebas y validación

6.1 Preparación de los datos de prueba

Para iniciar las pruebas del rendimiento del modelo, fue esencial extraer los datos necesarios que se utilizarían para evaluar los resultados. Este proceso involucró la recuperación de información de las bases de datos, las consultas en lenguaje natural y la creación de objetos de tipo query correspondientes a cada consulta, que se utilizarían como entrada para las pruebas.

6.1.1 Proceso de preparación con los datos recuperados SPIDER

Como se mencionó anteriormente, el dataset utilizado para entrenar y evaluar este modelo fue obtenido de SPIDER. A continuación, se describen los pasos realizados para recuperar la información necesaria y llevar a cabo las pruebas de eficacia del modelo:

1. Extracción de las bases de datos de prueba

En primer lugar, se llevó a cabo la extracción de algunas características de las bases de datos seleccionadas de Spider, como sus nombres, tablas y columnas. Esta etapa fue crucial, ya que estas características serán utilizadas posteriormente en la evaluación semántica que realiza el modelo.

2. Extracción de las consultas en español

En segundo lugar, se procedió a la extracción de las consultas en lenguaje natural, que también servirían como entrada al modelo. Estas preguntas fueron seleccionadas bajo el criterio de ser de nivel 1, es decir, consultas que siguen la estructura SQL estándar de tipo SELECT FROM WHERE y que pertenecen a las bases de datos seleccionadas.

3. Extracción de las consultas en formato de objeto Query

Finalmente, un paso crucial fue transformar las características de la consulta proporcionadas por Spider en objetos de tipo Query. Esto se debió a que el formato original de las partes de la consulta SQL en Spider no era directamente utilizable. Por lo tanto, se llevó a cabo un procesamiento adicional para mapear estas características a objetos de tipo Query para cada una de las preguntas. Estos objetos de tipo Query están compuestos por el nombre de la tabla, una lista de columnas y su respectiva condición, lo que facilita su posterior procesamiento y evaluación en el modelo.

En la Tabla III se detalla la cantidad de información que se recuperó:

<p>Tabla III</p> <p>Detalles de los datos de entrada provenientes de SPIDER para el rendimiento del modelo</p>	
Tipo	Cantidad
Objetos Database	166
Objetos Query	674
Consultas en lenguaje natural	674

6.1.1 Proceso de preparación con datos controlados

Para este caso, realizamos manualmente la instanciación de un objeto Database, definiendo sus tablas y columnas de manera personalizada. Además, creamos una consulta en lenguaje natural y su correspondiente objeto Query para llevar a cabo las pruebas.

A continuación, se presenta la definición de esta prueba:

Para Database:

<p>Tabla IV</p> <p>Detalles de los datos de entrada controlado para el rendimiento del modelo</p>		
Base de datos	Consulta SQL tipo Query	Consulta en lenguaje natural
<p>Nombre de la base: Database</p> <p>Tabla 1: Estudiantes</p> <p>Columnas:</p> <ul style="list-style-type: none"> • identificador • nombre • país • edad <p>Tabla 2: Cursos</p> <p>Columnas:</p>	<p>SELECT nombre, identificador WHERE pais = 'Mexico'</p>	<p>"Muestra todos los nombres de los alumnos y sus identificadores que estudian en México"</p>

<ul style="list-style-type: none"> • identificador • nombre • profesor 		
---	--	--

6.2 Validar el rendimiento del modelo

6.2.1 Evaluación Top-N Accuracy

Para obtener el porcentaje de Accuracy de nuestro modelo implementamos Top-N Accuracy auxiliado por Logical Form Accuracy. Para nuestro modelo en específico no utilizamos un número N de predicciones con mayor probabilidad en específico, utilizamos todas las predicciones generadas por el modelo para cada pregunta a evaluar. Tomamos por correcta una predicción si al menos una de las consultas SQL generadas por el modelo es igual a la consulta en SQL correcta.

Para comparar las consultas en SQL utilizamos los objetos Query, este objeto contiene los atributos de tabla, columnas y condición. La comparación se realiza con los respectivos atributos de cada Query, si contienen los mismos valores en los atributos las consideramos iguales. En el caso de las columnas al poder ser N columnas en la consulta SQL, solo consideramos que las consultas contengan el mismo número de columnas y las mismas columnas dentro del objeto Query, sin importar el orden en que las columnas se encuentren.

6.3 Resultados

6.3.1 Resultados obtenidos

Después, de configurar el entorno de nuestra validación y evaluación, los resultados que obtuvimos son los siguientes:

<p>Tabla V</p> <p>Resultados por experimento con los datos provenientes de SPIDER</p>					
Id experimento	Umbral de similitud	Umbral de similitud en condiciones	No. consultas correctamente predichas	No. total de consultas a prueba	Top-N Accuracy
1	-	-	0	674	0%
2	0.8	0.8	98	674	14.54%
3	0.8	0.5	131	674	19.43%
4	0.8	0.1	173	674	25.81%

<div> <div>Tabla VI</div> <div>Resultado para el experimento con datos controlados</div> </div>			
Id experimento	Tubería	Umbral (si aplica)	¿Predicción correcta?
1	TextPipeline	-	No
2	EmbeddingPipeline	0.6	Si

Para poder tener una mejor visión de los resultados obtenidos en nuestro experimento con mejor accuracy utilizamos la distancia Levenshtein. Esta distancia fue utilizada para comparar los atributos de las consultas reales con las consultas generadas por nuestro modelo. En la figura 6.1 se observan los resultados en gráficas de barras, dónde en el eje X se muestra la distancia existente entre la consulta real y la generada por el modelo, mientras en el eje Y se encuentra el número de consultas generadas por el modelo que se encuentran a las distancias indicadas por el eje X.

En palabras más simples, el eje X muestra el número de cambios que son necesarios en las consultas generadas por el modelo para llegar a ser idénticas a las consultas reales y en el eje Y el número de consultas que requieren ese número de cambios.

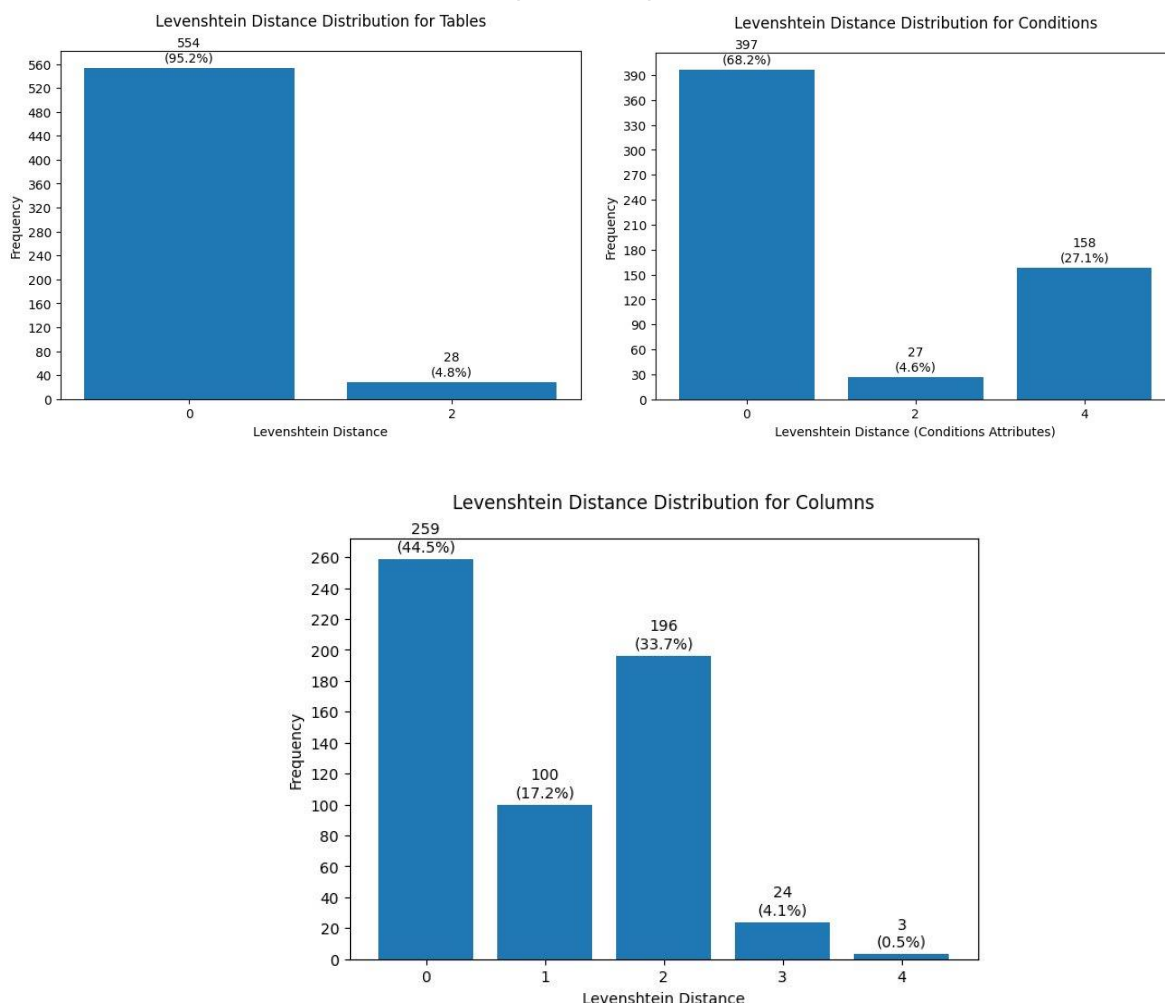


Fig 6.1. Distancia Levenshtein del experimento 4 tabla V

Ya calculados estos resultados, en el siguiente apartado analizaremos a detalle lo que nuestros hallazgos sugieren.

6.3.2 Análisis

Los resultados del experimento revelaron una baja precisión en las consultas generadas, lo que sugiere una falta de comprensión del modelo sobre la información implícita en las consultas en lenguaje natural. Particularmente, se observó una dependencia crítica en la definición del umbral en la tubería de embedding, lo que destaca la importancia de ajustar este parámetro para mejorar el rendimiento del sistema. Además, se identificó que en entornos más controlados, con la capacidad de manipular características de la base de datos y ajustar el umbral, el modelo podría capturar de manera más precisa la información necesaria para generar consultas SQL. Estos hallazgos subrayan la necesidad de mejorar la capacidad del modelo para interpretar y procesar consultas con mayor variabilidad, proporcionando una base sólida para futuras investigaciones y mejoras en sistemas de procesamiento de lenguaje natural para la generación de consultas SQL.

CONCLUSIONES

Se ha logrado implementar un modelo generativo de SQL a partir de consultas en español, mostrando que es posible utilizar el lenguaje natural como herramienta para generar consultas básicas en SQL. La investigación exhaustiva de las estructuras sintácticas y semánticas de SQL permitió desarrollar reglas precisas que mejoraron la efectividad del modelo. Se construyó un conjunto de ejemplos etiquetados, esenciales para el entrenamiento del modelo, que incluyeron consultas en español, esquemas de bases de datos, y sus correspondientes representaciones en SQL.

Adicionalmente, se seleccionó y desarrolló una arquitectura capaz de capturar patrones complejos de dependencias entre palabras y partes de la frase en español, lo cual resultó en una traducción más precisa a consultas SQL. Diversos experimentos se realizaron para asegurar la efectividad y corrección del modelo, validando los resultados obtenidos. Aunque se logró cumplir parcialmente con este objetivo, se identificaron áreas de mejora que podrían optimizar aún más el modelo.

Por lo tanto, los productos resultantes incluyen un conjunto de ejemplos etiquetados para entrenar el modelo, una arquitectura para capturar patrones lingüísticos complejos y un modelo de IA que genera consultas SQL a partir de consultas en español. Estos avances representan un paso significativo hacia la integración de lenguajes naturales en la generación automática de consultas SQL, facilitando el acceso y manejo de bases de datos a usuarios no técnicos.

TRABAJO A FUTURO

El trabajo futuro se centrará en explorar nuevas estrategias y técnicas para mejorar el rendimiento del modelo generativo de SQL a partir de consultas en español. Para lograr esto, se implementarán diversas acciones y mejoras, tales como:

1. **Implementar nuevas técnicas de procesamiento de texto:** Se explorarán y adoptarán técnicas avanzadas de procesamiento de lenguaje natural para mejorar la comprensión y el manejo del texto en español, optimizando así la generación de consultas SQL.
2. **Refinar y generar reglas más efectivas:** Se investigarán y desarrollarán estrategias para la creación de reglas más precisas y efectivas, superando las limitaciones de las reglas actuales y mejorando la precisión del modelo.
3. **Agregar un extractor de secciones con reglas específicas:** Se implementará un extractor de secciones basado en reglas para identificar palabras clave que indiquen la presencia de agregadores y agrupadores, dotando al modelo de la capacidad de manejar consultas de Nivel 2.
4. **Desarrollar una herramienta educativa:** Se creará un sistema que permita utilizar el modelo como una herramienta para el aprendizaje, facilitando la enseñanza y el entendimiento de la generación de consultas SQL a partir del lenguaje natural.
5. **Desarrollar una métrica de ranking/posicionamiento:** Se establecerá una métrica para evaluar y rankear las consultas generadas, indicando las mejores opciones y mejorando la selección de las consultas más precisas y relevantes.

Estas mejoras no solo incrementarán la precisión del modelo, sino que también ampliarán su capacidad para manejar consultas más complejas, elevando su aplicabilidad en diversos contextos.

REFERENCIAS

- [1] D. D. Chamberlin, “Early History of SQL”, *IEEE Ann. Hist. Comput.*, vol. 34, núm. 4, pp. 78–82, oct. 2012, doi: 10.1109/MAHC.2012.61.
- [2] W. A. Woods, “Progress in natural language understanding: an application to lunar geology”, en *Proceedings of the June 4-8, 1973, national computer conference and exposition on - AFIPS '73*, New York, New York: ACM Press, 1973, p. 441. doi: 10.1145/1499586.1499695.
- [3] D. Alconada, “AlcoNQL : Herramienta de consulta SQL por medio de lenguaje natural”, Universitat Oberta de Catalunya, 2013. [En línea]. Disponible en: traductor
- [4] M. Bonilla, “Traductor de consultas del lenguaje natural a SQL”, Universidad Central “Marta Abreu” de Las Villas, 2011. [En línea]. Disponible en: https://dspace.uclv.edu.cu/bitstream/handle/123456789/9225/%5b06-28%5d%20Trabajo%20de%20Diploma_Marlen%20FINAL%20OK%20.pdf?sequence=1&isAllowed=y
- [5] R. Cai, J. Yuan, B. Xu, y Z. Hao, “SADGA: Structure-Aware Dual Graph Aggregation Network for Text-to-SQL”. arXiv, el 17 de enero de 2022. Consultado: el 23 de abril de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/2111.00653>
- [6] Y. Mellah, A. Rhouati, E. H. Ettifouri, T. Bouchentouf, y M. G. Belkasmi, “SQL Generation from Natural Language: A Sequence-to-Sequence Model Powered by the Transformers Architecture and Association Rules”, *J. Comput. Sci.*, vol. 17, núm. 5, pp. 480–489, may 2021, doi: 10.3844/jcssp.2021.480.489.
- [7] D. Reinsel, J. Gantz, y J. Rydning, “The Digitization of the World from Edge to Core”, 2018.
- [8] A. Bautista Alvarado y others, “Traducción de consultas del lenguaje natural español a SQL que involucran agrupamiento”, 2014.
- [9] B. Qin *et al.*, “A Survey on Text-to-SQL Parsing: Concepts, Methods, and Future Directions”. arXiv, el 29 de agosto de 2022. Consultado: el 12 de abril de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/2208.13629>
- [10] L. Ouyang *et al.*, “Training language models to follow instructions with human feedback”, arXiv.org. Consultado: el 12 de noviembre de 2023. [En línea]. Disponible en: <https://arxiv.org/abs/2203.02155v1>
- [11] A. Liu, X. Hu, L. Wen, y P. S. Yu, “A comprehensive evaluation of ChatGPT’s zero-shot Text-to-SQL capability”. arXiv, el 11 de marzo de 2023. Consultado: el 23 de abril de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/2303.13547>
- [12] S. Studer *et al.*, “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology”. arXiv, el 24 de febrero de 2021. doi: 10.48550/arXiv.2003.05155.
- [13] “ml-ops.org”. Consultado: el 14 de abril de 2023. [En línea]. Disponible en: <https://ml-ops.org/>
- [14] A. Kumar, P. Nagarkar, P. Nalhe, y S. Vijayakumar, “Deep Learning Driven Natural Languages Text to SQL Query Conversion: A Survey”, arXiv.org. Consultado: el 12 de noviembre de 2023. [En línea]. Disponible en: <https://arxiv.org/abs/2208.04415v1>
- [15] G. Katsogiannis-Meimarakis y G. Koutrika, “Deep Learning Approaches for Text-to-SQL Systems”. OpenProceedings.org, 2021. doi: 10.5441/002/EDBT.2021.90.
- [16] T. Yu, Z. Li, Z. Zhang, R. Zhang, y D. Radev, “TypeSQL: Knowledge-Based Type-Aware Neural Text-to-SQL Generation”, en *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 588–594. doi: 10.18653/v1/N18-2093.
- [17] V. Zhong, C. Xiong, y R. Socher, “Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning”. arXiv, el 9 de noviembre de 2017. Consultado: el 12 de abril de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/1709.00103>
- [18] X. Xu, C. Liu, y D. Song, “SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning”. arXiv, el 13 de noviembre de 2017. Consultado: el 11 de

- abril de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/1711.04436>
- [19] T. Yu *et al.*, “Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task”. arXiv, el 2 de febrero de 2019. Consultado: el 16 de noviembre de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/1809.08887>
- [20] U. Brunner y K. Stockinger, “ValueNet: A Natural Language-to-SQL System that Learns from Database Information”. arXiv, el 22 de febrero de 2021. Consultado: el 12 de abril de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/2006.00888>
- [21] B. Wang, R. Shin, X. Liu, O. Polozov, y M. Richardson, “RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers”. arXiv, el 24 de agosto de 2021. Consultado: el 19 de agosto de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/1911.04942>
- [22] H. Fu, C. Liu, B. Wu, F. Li, J. Tan, y J. Sun, “CatSQL : Towards Real World Natural Language to SQL Applications”, *Proc. VLDB Endow.*, vol. 16, núm. 6, pp. 1534–1547, feb. 2023, doi: 10.14778/3583140.3583165.
- [23] F. Reyes García, “LNE2SQL: traductor de consultas del lenguaje natural a SQL v2.0”, Universidad Central “Marta Abreu” de Las Villas, 2012. [En línea]. Disponible en: <https://dspace.uclv.edu.cu/bitstream/handle/123456789/6067/Frank%20Reyes%20Garcia--Tesis.pdf?sequence=1&isAllowed=y>
- [24] A. Meier y M. Kaufmann, *SQL & Nosql Databases: Models, Languages, Consistency Options and Architectures for Big Data Management*. New York, NY, 2019.
- [25] “(I.B.D.) FUNDAMENTOS DE SQL (3ª ED.) | SHELDON OPPEL | Segunda mano | McGraw-Hill Interamericana de España S.L. | Casa del Libro México”, *casadellibro*. Consultado: el 2 de abril de 2024. [En línea]. Disponible en: <https://latam.casadellibro.com/libro-ibd-fundamentos-de-sql-3-ed/9786071502513/1867787>
- [26] B. Mahesh, *Machine Learning Algorithms -A Review*. 2019. doi: 10.21275/ART20203995.
- [27] E. Saliba, “An overview of Pattern Recognition”.
- [28] B. Arinze, “Selecting appropriate forecasting models using rule induction”, *Omega*, vol. 22, núm. 6, pp. 647–658, nov. 1994, doi: 10.1016/0305-0483(94)90054-X.
- [29] P. Flach y N. Lavrač, “Rule Induction”, en *Intelligent Data Analysis*, M. Berthold y D. J. Hand, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 229–267. doi: 10.1007/978-3-540-48625-1_7.
- [30] G. Cervone, P. Franzese, y A. P. K. Keese, “Algorithm quasi-optimal (AQ) learning”, *WIREs Comput. Stat.*, vol. 2, núm. 2, pp. 218–236, mar. 2010, doi: 10.1002/wics.78.
- [31] A. C. Vasquez, “Procesamiento de lenguaje natural”, *Rev. Investig. Sist. E Inform.*, ene. 2009, Consultado: el 12 de noviembre de 2023. [En línea]. Disponible en: https://www.academia.edu/66213908/Procesamiento_de_lenguaje_natural
- [32] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, y D. Brown, “Text Classification Algorithms: A Survey”, *Information*, vol. 10, núm. 4, Art. núm. 4, abr. 2019, doi: 10.3390/info10040150.
- [33] “How do you design and implement NLP pipelines and workflows?” Consultado: el 24 de abril de 2024. [En línea]. Disponible en: <https://www.linkedin.com/advice/0/how-do-you-design-implement-nlp-pipelines>
- [34] “What is Data Pipeline? - Data Pipeline Explained - AWS”, Amazon Web Services, Inc. Consultado: el 24 de abril de 2024. [En línea]. Disponible en: <https://aws.amazon.com/what-is/data-pipeline/>
- [35] “What Is a Machine Learning Pipeline? | IBM”. Consultado: el 24 de abril de 2024. [En línea]. Disponible en: <https://www.ibm.com/topics/machine-learning-pipeline>
- [36] P. Khare, “Deep Learning for NLP: Word2Vec, Doc2Vec, and Top2Vec Demystified”, *MLearning.ai*. Consultado: el 12 de noviembre de 2023. [En línea]. Disponible en: <https://medium.com/mlearning-ai/deep-learning-for-nlp-word2vec-doc2vec-and-top2vec-demystified-3842b4fad5c9>
- [37] J. Thanaki, *Python Natural Language Processing*. Packt Publishing Ltd, 2017.
- [38] M. T. Pilehvar y J. Camacho-Collados, *Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning*. Springer Nature, 2022.

- [39]N. Deng, Y. Chen, y Y. Zhang, “Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect”. arXiv, el 22 de agosto de 2022. Consultado: el 16 de noviembre de 2023. [En línea]. Disponible en: <http://arxiv.org/abs/2208.10099>
- [40]F. Özcan, A. Quamar, J. Sen, C. Lei, y V. Efthymiou, “State of the Art and Open Challenges in Natural Language Interfaces to Data”, en *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, Portland OR USA: ACM, jun. 2020, pp. 2629–2636. doi: 10.1145/3318464.3383128.
- [41]T. Saadani, “Understanding Data Labels and Data Labeling: Definition, Types, and How it Works for Machine...”, Medium. Consultado: el 2 de abril de 2024. [En línea]. Disponible en: <https://medium.com/@takouasaadani/understanding-data-labels-and-data-labeling-definition-types-and-how-it-works-for-machine-b7c85b52c593>
- [42]Admin, “A Brief Guide about the Data Annotation”. Consultado: el 2 de abril de 2024. [En línea]. Disponible en: <https://macgence.com/blog/a-brief-guide-about-the-data-annotation/>
- [43]“TopN Accuracy-where to use & how to calculate?” Consultado: el 15 de mayo de 2024. [En línea]. Disponible en: <https://www.linkedin.com/pulse/topn-accuracy-where-use-how-calculate-chandra-sharat>
- [44]“Introduction to Levenshtein distance”, GeeksforGeeks. Consultado: el 9 de junio de 2024. [En línea]. Disponible en: <https://www.geeksforgeeks.org/introduction-to-levenshtein-distance/>
- [45]L. Olmos Camarena, *Aplicaciones del Procesamiento de Lenguaje Natural y Recuperación de la Información*. 2019.
- [46]Sommerville, *Ingeniería De Software*. 2011.
- [47]J. L. Moya, A. M. Becerra, y C. A. Chagoyén, “Utilización de Sistemas Basados en Reglas y en Casos para diseñar transmisiones por tornillo sinfín”, . *ISSN*, vol. 15, núm. 1.
- [48]S. Navarro, “Similitud entre vectores o cosine similarity”. Consultado: el 9 de junio de 2024. [En línea]. Disponible en: <https://keepcoding.io/blog/similitud-entre-vectores-o-cosine-similarity/>

GLOSARIO

1. AQ (Algoritmo Quasi-Optimal): Algoritmo de aprendizaje automático basado en reglas para la clasificación de datos, que genera reglas explícitas que describen cómo clasificar diferentes ejemplos en función de los valores de sus atributos.
2. Base de datos: Conjunto organizado de datos almacenados y accesibles electrónicamente. Las bases de datos pueden ser manipuladas y consultadas utilizando sistemas de gestión de bases de datos (DBMS) y lenguajes como SQL.
3. Clasificación: Proceso de asignar una etiqueta o categoría a un texto en función de las características identificadas utilizando técnicas como la clasificación binaria, multinomial o el aprendizaje profundo.
4. Condición: Parte de una consulta SQL que especifica los criterios para seleccionar los datos a ser recuperados.
5. Consulta SQL: Instrucción en SQL utilizada para realizar operaciones en una base de datos, como recuperación, inserción, actualización o eliminación de datos.
6. Distancia de Levenshtein: Métrica de distancia que mide el número de ediciones mínimas necesarias para transformar una cadena de caracteres en otra. Las operaciones permitidas son inserciones, eliminaciones o sustituciones de un solo carácter.
7. Embedding: Representaciones vectoriales de palabras que capturan su significado semántico en un espacio multidimensional, generadas mediante técnicas como Word2Vec, GloVe o FastText.
8. Evaluador Semántico: Componente de un sistema que valida la semántica de las consultas generadas para asegurar que sean coherentes y correctas.
9. Extracción de Características: Identificación de atributos relevantes en el texto que pueden utilizarse para distinguir entre clases, como la frecuencia de palabras o la presencia de términos clave.
10. Inducción de Reglas: Método en aprendizaje automático que descubre patrones significativos en conjuntos de datos mediante la construcción de reglas lógicas que representan relaciones y regularidades presentes en la información.
11. Mapeo a Plantilla SQL: Proceso de convertir una consulta en lenguaje natural en una consulta SQL utilizando una plantilla predefinida.
12. Normalización: Transformación de palabras a una forma estándar para reducir la variabilidad.
13. PLN (Procesamiento de Lenguaje Natural): Campo de estudio en Inteligencia Artificial que se enfoca en permitir que las computadoras comprendan, interpreten y generen lenguaje humano de manera efectiva.
14. Regla: Condición lógica generada por un algoritmo que describe cómo clasificar ejemplos en función de los valores de sus atributos.
15. Sección: Parte de una consulta en lenguaje natural que potencialmente contiene información relevante para formar una consulta SQL.
16. Semilla: Ejemplo positivo seleccionado aleatoriamente en el algoritmo AQ para generar reglas.
17. Tokenización: Proceso de dividir el texto en unidades más pequeñas llamadas tokens, que pueden ser palabras individuales, símbolos de puntuación o caracteres.

18. Tuberías de Procesamiento (PLN pipelines): Serie de pasos que transforman datos de texto sin procesar en una salida deseada, como una etiqueta, resumen o respuesta, organizando y optimizando el flujo de trabajo de procesamiento del lenguaje natural.
19. Vectorización: Representación de palabras como vectores numéricos en un espacio vectorial para facilitar el procesamiento mediante algoritmos de aprendizaje automático.
20. Stemming: Proceso de reducir las palabras a su raíz o base eliminando prefijos y sufijos, sin garantizar que la palabra resultante sea una palabra real en el idioma.
21. Stop words: Palabras comunes en un idioma que generalmente se eliminan durante el procesamiento de texto porque no aportan información significativa para tareas específicas.
22. Text to SQL: Tarea de convertir texto en lenguaje natural a consultas SQL estructuradas.
23. Top-N Accuracy: Métrica de evaluación utilizada para medir la precisión de un modelo, calculando la proporción de ejemplos en los que la respuesta correcta se encuentra entre las N principales predicciones del modelo.
24. Vector de Características: Representación de un conjunto de características o atributos en forma de vector numérico, utilizado como entrada para algoritmos de aprendizaje automático.
25. Word Embeddings: Técnicas que asignan vectores densos y de baja dimensionalidad a palabras, capturando sus relaciones semánticas y contextuales.

APÉNDICES

Apéndice 1. Índice de figuras

Fig 3.1. Formulando una consulta en SQL.....	21
Fig 3.2. Algoritmo AQ.....	27
Fig 3.3. Algoritmo de generación de estrella.....	28
Fig 3.4. Paso 0: Cargar el conjunto de datos y dividirlo en 2 clases.....	28
Fig 3.5. Obtención de primera regla.....	29
Fig 3.6. Obtención de segunda regla.....	29
Fig 3.7. Obtención de tercera regla.....	30
Fig 3.8. Visualización de ejemplos positivos completamente abarcados.....	30
Fig 3.9. Ejemplo Top-N Accuracy.....	38
Fig 4.1. Diagrama de la arquitectura que proponemos, la cual llamamos ñ2SQL.....	41
Fig 4.2. Diagrama de la arquitectura haciendo énfasis en la extracción de secciones.....	41
Fig 4.3. Diagrama de la arquitectura haciendo énfasis en el proceso de tratamiento.....	42
Fig 4.4. Diagrama de la arquitectura haciendo énfasis en el mapeo a la plantilla SQL.....	42
Fig 4.5. Diagrama de la arquitectura haciendo énfasis en el evaluador semántico.....	43
Fig 5.1. Estructura dataset Spider.....	48
Fig 5.2. Estructura TRAIN_SPIDER.json.....	49
Fig 5.3. Etiquetados realizados.....	50
Fig 5.4. Resultados del entrenamiento de nuestra implementación del algoritmo AQ para obtener los contextos de las secciones principales.....	52
Fig 5.5. Reglas para la clase ATRIBUTO.....	53
Fig 5.6. Paso 1: Entradas.....	57
Fig 5.7. Paso 2: Instanciación de las reglas a utilizar.....	58
Fig 5.8. Paso 3: Configuración de las tuberías de datos.....	58
Fig 5.9. Paso 4: Extracción de secciones.....	58
Fig 5.10. Paso 5: Limpieza de secciones.....	59
Fig 5.11. Paso 6: Generación de posibles tripletas.....	59
Fig 5.12. Paso 7: Validación de consultas a través del evaluador semántico.....	60
Fig 5.13. Paso 8: Obtención de consultas finales.....	60
Fig 6.1. Distancia Levenshtein del experimento 4 tabla V.....	64
Fig A1. Boceto para la interfaz IU1 Consulta en Lenguaje Natural a SQL.....	75
Fig A2. Diagrama de clases del sistema.....	76
Fig A3. Diagrama de secuencia del flujo principal.....	76

Apéndice 2. Índice de tablas

Tabla I	
Análisis detallado de los costos asociados en la elaboración del proyecto	44
Tabla II	
Análisis de riesgos asociados al desarrollo del proyecto.....	45
Tabla III	
Detalles de los datos de entrada provenientes de SPIDER para el rendimiento del modelo	61
Tabla IV	
Detalles de los datos de entrada controlado para el rendimiento del modelo	61
Tabla V	
Resultados por experimento con los datos provenientes de SPIDER	62
Tabla VI	
Resultado para el experimento con datos controlados.....	63
Tabla A1	
Comportamiento de acciones para el CU1 Consulta en Lenguaje Natural	74

Apéndice 3: Posible aplicación práctica del modelo como parte de un sistema

En el proceso de desarrollo del trabajo terminal, se identificó la posibilidad de proponer la incorporación de una funcionalidad adicional que inicialmente no está contemplada en la planificación del proyecto; sin embargo, como el modelo finalizado tiene como objetivo servir como una herramienta para aquellos que recién comienzan a aprender SQL, es entonces que, a continuación, se presenta la descripción de un posible caso de uso del modelo dentro de un sistema web y su correspondiente especificación, como el diagrama de clases, diagrama de secuencia:


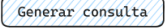
1. CU1 Consulta en Lenguaje Natural a SQL

1.1. Resumen

Permite a un usuario general formular consultas en lenguaje natural en español y obtener consultas SQL válidas para ser ejecutadas en una base de datos. El sistema traduce las consultas del usuario y proporciona los resultados obtenidos de la base de datos.

1.2. Trayectorias del Caso de Uso

1.2.1. Trayectoria principal

1. El sistema muestra la interfaz IU1 Consulta en Lenguaje Natural.
2. El usuario presiona el botón  de la interfaz IU1 Consulta en Lenguaje Natural.
3. Abre la ventana del navegador de archivos del equipo que se esté utilizando. [Trayectoria A]
4. El usuario selecciona el archivo que tenga el esquema de su base de datos .
5. Verifica que el formato del archivo seleccionado sea válido. [Trayectoria B]
6. El usuario ingresa su consulta en lenguaje natural en español.
7. El usuario presiona el botón .
8. El sistema verifica que los campos marcados como obligatorios hayan sido ingresados. [Trayectoria C]
9. El sistema procesa la consulta y la traduce a una consulta SQL.
10. El sistema muestra el resultado obtenido por el modelo, ya sea la consulta SQL generada, o una abstención.

- - - - Fin del caso de uso.

1.2.2. Trayectoria alternativa A

Condición: El usuario desea cambiar el archivo.

1. El usuario presiona el botón 

2. Continúa en el paso 4 de la trayectoria principal del caso de uso

- - - - Fin de la trayectoria.

1.2.3. Trayectoria alternativa B

Condición: El formato del archivo no es válido.

1. El sistema muestra el mensaje: “Favor de ingresar un archivo válido.”

2. Continúa en el paso 2 de la trayectoria principal del caso de uso

- - - - Fin de la trayectoria.

1.2.4. Trayectoria alternativa C

Condición: El actor no ingresó uno o más campos obligatorios.

3. El sistema muestra el mensaje: “Este campo es obligatorio. Favor de ingresar la información.”



4. Continúa en el paso 2 de la trayectoria principal del caso de uso

- - - - Fin de la trayectoria.

1.3. Diseño de la interfaz

En la Figura A1 se muestra la interfaz IU1, correspondiente a la funcionalidad descrita en la trayectorias, y en la Tabla A1 se muestran las acciones que se pueden realizar en esta interfaz.

Tabla A1
Comportamiento de acciones para el CU1 Consulta en Lenguaje Natural

Acciones	
Acción:	Cargar archivo al sistema.
Representación:	
Descripción:	<ul style="list-style-type: none">• Permite al actor seleccionar un archivo al sistema.
Condición:	Ninguna.
Acción:	Generar consulta SQL
Representación:	
Descripción:	<ul style="list-style-type: none">• Muestra la consulta generada
Condición:	Ninguna

The wireframe is titled "ñ2SQL" in a large, bold font. Below the title is a horizontal line. The interface is divided into three main sections, each with a numbered instruction:

- * 1. Ingrese el esquema de su base de datos**: This section contains a text input field with a file upload icon (a document with a plus sign) on the right.
- * 2. Ingrese su consulta en español**: This section contains a text input field with the example query: "Muestra toda la información sobre los empleados que trabajan en México". Below this input field is a button labeled "Generar consulta".
- 3. SQL**: This section contains a text input field with the placeholder text: "Aquí observará su consulta en SQL".

Fig A1. Boceto para la interfaz IU1 Consulta en Lenguaje Natural a SQL

Nota: Esta funcionalidad adicional se considera un agregado al trabajo terminal, debido a que se encuentra fuera del alcance del mismo. Su implementación dependerá de la disponibilidad de tiempo durante el desarrollo del proyecto.

Diagrama de clases

El diagrama de clases es una representación visual que muestra la estructura estática del sistema, identificando las clases, sus atributos y métodos, así como las relaciones entre ellas [46].

El propósito principal de este diagrama es mostrar de manera visual la estructura de clases y cómo están interconectadas en el sistema para permitir la ejecución exitosa del caso de uso. Detalla las clases, sus propiedades y métodos, así como las relaciones de asociación, herencia o dependencia entre ellas.

Mediante este diagrama, se busca ofrecer una comprensión clara y detallada de la estructura de clases y la organización interna del sistema en relación con el caso de uso específico, lo que facilita la visualización de cómo interactúan las diferentes entidades para lograr la funcionalidad deseada.

El diagrama de clases propuesto para abordar este modelo como sistema puede verse en la figura A2

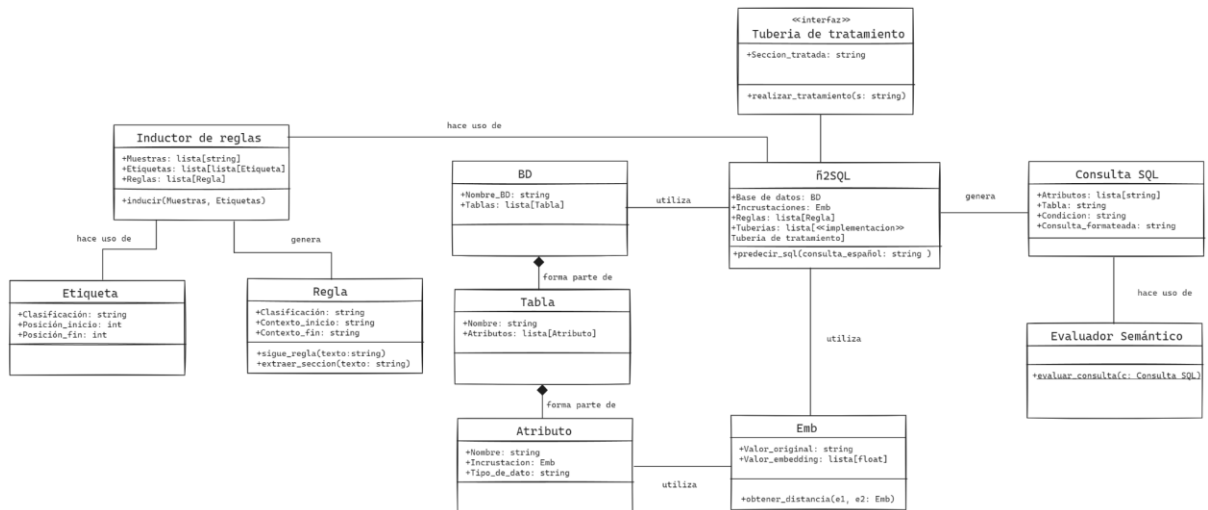


Fig A2. Diagrama de clases del sistema.

Diagrama de secuencia

El diagrama de secuencia es una representación visual que muestra la interacción entre diferentes objetos y clases a lo largo del proceso de ejecución del caso de uso [46].

El objetivo principal de este diagrama es ilustrar de manera secuencial las interacciones entre los objetos involucrados durante las distintas etapas del caso de uso, detallando cómo se comunican entre sí y cómo se llevan a cabo las acciones requeridas para procesar una consulta en lenguaje natural y obtener una consulta SQL correspondiente [46].

A través de este diagrama, se busca brindar una comprensión clara y visual de cómo ocurren las interacciones entre las clases y los objetos durante la ejecución del caso de uso, lo que permite una mejor comprensión del flujo de trabajo y la lógica detrás del proceso de traducción de consultas.

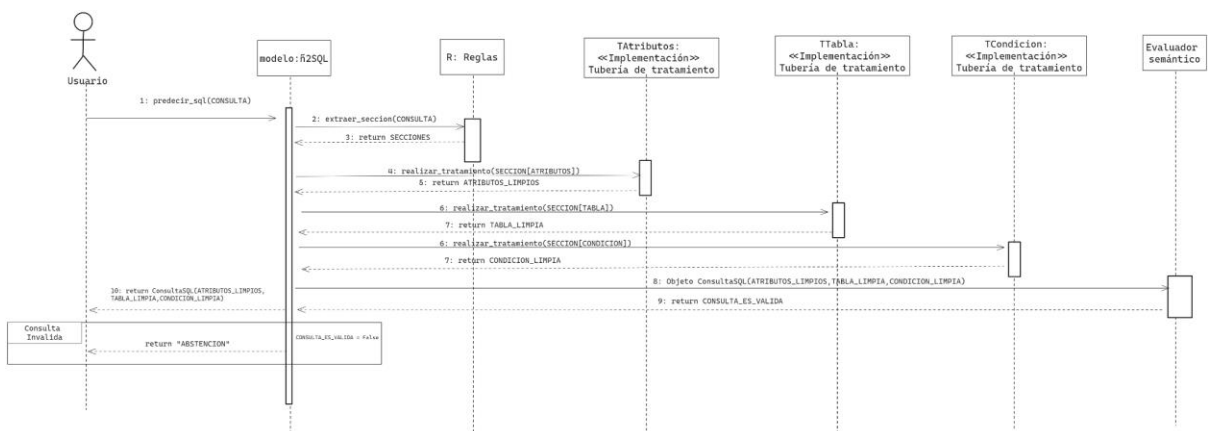


Fig A3. Diagrama de secuencia del flujo principal.