


Extracción de conocimiento mediante traducción automatizada a Cypher con aprendizaje *zero-shot*

Knowledge extraction through automated translation to Cypher with zero-shot learning

Rolando Sánchez Ramos^{1*} 

Resumen Este trabajo se centra en abordar la complejidad inherente a la consulta de bases de datos en forma de grafo, como Neo4J. Estas bases de datos a menudo requieren un conocimiento especializado en lenguajes de consulta, lo que limita su accesibilidad a un grupo reducido de usuarios con habilidades técnicas avanzadas. Para superar esta limitación, proponemos la aplicación del aprendizaje con cero muestras de entrenamiento (*zero-shot*), un enfoque innovador en el procesamiento del lenguaje natural. En esta investigación, se lleva a cabo un experimento basado en el modelo GPT-4 para traducir consultas de lenguaje natural a código Cypher. La evaluación se realiza utilizando el conjunto de datos de evaluación MetaQA, que abarca una amplia variedad de ejemplos de consultas. Los resultados obtenidos fueron del 76,53 %, 43,45 % y 31,03 % para los tres lotes de evaluación del *benchmark* utilizado, mejorando de esta forma el mejor resultado de modelos de lenguaje en la traducción de lenguaje natural a código Cypher sobre MetaQA mediante el aprendizaje *zero-shot*.

Palabras Clave: aprendizaje con cero muestras de entrenamiento, modelos de lenguajes a gran escala, inteligencia artificial

Abstract This work focuses on addressing the inherent complexity of querying graph databases, such as Neo4J. These databases often require specialized knowledge in query languages, limiting their accessibility to a small group of users with advanced technical skills. To overcome this limitation, we propose the application of zero-shot learning, an innovative approach in natural language processing. In this research, an experiment is conducted based on the GPT-4 model to translate natural language queries into Cypher code. The evaluation is carried out using the MetaQA evaluation dataset, which covers a wide variety of query examples. The results obtained were 76,53 %, 43,45 %, and 31,03 % for the three evaluation lots of the benchmark used, thereby improving the best result of language models in translating natural language into Cypher code using zero-shot learning.

Keywords: zero-shot learning, large language models, artificial intelligence

Mathematics Subject Classification: MSC68

¹Departamento de Computación, Facultad de Matemática y Computación, Universidad de La Habana, Cuba. Email: rolsanchez@yandex.com

*Autor para Correspondencia (Corresponding Author)

Editado por (Edited by): Damian Valdés Santiago, Facultad de Matemática y Computación, Universidad de La Habana, La Habana, Cuba.

Citar como: Sánchez Ramos, R. (2024). Extracción de conocimiento mediante traducción automatizada a Cypher con aprendizaje *zero-shot*. *Ciencias Matemáticas*, 37(1), 61–68. DOI: <https://doi.org/10.5281/zenodo.14876950>. Recuperado a partir de <https://revistas.uh.cu/rcm/article/view/9192>.

Introducción

En la época actual, asistimos a un constante aumento en la producción de información en diversos formatos: visual, auditivo y textual, que abarca todos los ámbitos de la sociedad [13].

De manera particular, resulta sumamente intrigante la información generada a través del ingenio creativo y la investigación humana. Estos tipos de datos se almacenan debido a su relevancia y a la necesidad de acceder a ellos en el futuro, pudiendo optar por una organización estructurada o no.

Las bases de conocimiento constituyen un tipo particular

de bases de datos diseñadas para la administración del saber. Estas bases brindan los medios para recolectar, organizar y recuperar digitalmente un conjunto de conocimientos, ideas, conceptos o datos [3]. La ventaja fundamental de mantener la información de manera estructurada radica en su facilidad para ser consultada, ampliada y modificada.

Debido a su utilidad y prevalencia, la recuperación de información a través de consultas en bases de conocimiento se ha convertido en una tarea esencial.

Es primordial que la información almacenada en bases de conocimiento adopte un formato adecuado para permitir búsquedas ágiles y precisas. Entre los formatos más comunes

se encuentran los modelos de Entidad-Relación y el modelo Relacional. A pesar de ser enfoques más antiguos, el modelo Relacional (BDR) sigue siendo el más ampliamente utilizado en la actualidad [10].

No obstante, en ocasiones, las características específicas del problema demandan un formato más expresivo, y es en este punto donde las bases de datos orientadas a grafos (BDOG) [22] entran en juego.

Las BDOG han ganado progresivamente popularidad como una manera efectiva de almacenar información en los últimos años. Estas bases tienen la capacidad de modelar una diversidad de situaciones del mundo real al tiempo que mantienen un alto nivel de simplicidad y legibilidad para los seres humanos. Las BDOG presentan numerosas ventajas en comparación con las bases de datos relacionales [22]. Esto incluye un mejor rendimiento, permitiendo el manejo más rápido y eficaz de grandes volúmenes de datos relacionados; flexibilidad, ya que la teoría de grafos en la que se basan las BDOG permite abordar diversos problemas y encontrar soluciones óptimas; y escalabilidad, ya que las bases de datos orientadas a grafos permiten una escalabilidad eficaz al facilitar la incorporación de nuevos nodos y relaciones entre ellos. Ejemplo de un sistema de gestión de BDOG es *Neo4J* [16], a través del cual es posible construir instancias de este tipo de base de datos e interactuar con las mismas a través del lenguaje de programación *Cypher* [17], el cual posee una sintaxis declarativa similar a SQL [21].

Por otro lado, el avance en la comprensión del lenguaje natural se ha visto potenciado con el surgimiento de los modelos de lenguajes a gran escala (LLM, por sus siglas en inglés) [20] como GPT-4 [14] o LLaMA-2 [7], los cuales presentan una serie de habilidades emergentes como elaboración de resúmenes de textos, generación de código, razonamiento lógico, entre otras [9].

Dichas herramientas constituyen modelos de *Machine Learning* entrenados con un gran volumen de datos, lo cual es posible gracias al número de parámetros con los que estos son configurados [20].

Usualmente, para el uso de los LLMs basta con ofrecerles como dato de entrada un texto [20], el cual describe la tarea que se espera que estos realicen. Además, son muchas las técnicas existentes para elaborar una entrada de calidad, esto con el objetivo de que la respuesta por parte de dicho modelo de lenguaje ofrezca resultados alentadores al respecto, lo cual se conoce como *prompt engineering* [20].

Una técnica bastante común es el aprendizaje con cero muestras de entrenamiento (*zero-shot learning*, ZSL, por sus siglas en inglés) [15], la cual consiste en describirle a un LLM un procedimiento a realizar sin ofrecer de antemano ejemplos de cómo resolverlo, como por ejemplo, en tareas relacionadas con la generación de código, donde algunos de estos son capaces de generar algoritmos expresados en un lenguaje de programación formal a partir de una sentencia o consulta en lenguaje natural sin recibir como entrada del usuario algunos ejemplos de código, o especificaciones de cómo funciona el

lenguaje objetivo a generar [11, 5].

Un problema bastante común con el uso de LLMs para la resolución de tareas como responder a consultas de conocimiento sin un contexto previo son las alucinaciones [8], las cuales provocan que, con cierta probabilidad, estos modelos produzcan como salida afirmaciones erróneas sobre hechos, personas o alguna referencia en el mundo real, lo cual resulta un inconveniente para situaciones donde se realicen consultas en lenguaje natural sobre un dominio reducido, como el referente a una base de conocimiento sobre un tema específico. Por lo tanto, no es suficiente el uso de solamente un LLM para responder a dichas preguntas, sino que también sería necesario que estos cuenten de antemano con una información referente al dominio específico sobre el cual se desea consultar.

En lo que respecta a la comprensión del lenguaje natural y su uso en consultas a bases de conocimiento, existen diversas vías llevadas a cabo y con resultados diversos, donde se hacen análisis sintácticos y semánticos sobre la consulta, muchas veces asistidos por diccionarios o mapas sobre la base de conocimiento en cuestión. Se usan modelos de paráfrasis como técnica de aumento de datos y finalmente *transformers* o incluso LLMs para llevar de la consulta ya curada al lenguaje de consulta formal o a un lenguaje intermedio capaz de expresar a esta a alto nivel, para finalmente ser ejecutada sobre un sistema de gestión de información [6, 4].

Problemática

Para extraer información procedente de una base de conocimiento basada en una BDOG generalmente, es necesario el uso de un lenguaje de consulta formal que pueda ser ejecutado sobre un sistema de gestión de esta. El enfoque más común para generar dicha consulta formal a partir de una consulta en lenguaje natural humano se basa en utilizar un modelo de aprendizaje automático entrenado para dicho proceso.

Sucede que para esta tarea en cuestión, es necesario elaborar ejemplos entrenantes de calidad y luego, ejecutar un proceso de entrenamiento, lo cual representa un importante gasto computacional que puede ser omitido con el uso de modelos ya entrenados para disímiles tareas y con capacidad de generar respuesta a preguntas para las cuales no fueron directamente preparados. Es por esta razón que se hace necesario el uso de un LLM, ya que estos poseen habilidades como la generación de código sin necesidad de ser reentrenados para ser utilizados efectivamente [20], y con ello, interactuar con una BDOG para extraer conocimiento.

Por otro lado, debido al problema que tienen los LLMs con el fenómeno de las alucinaciones, se hace necesario encontrar una vía de darle como entrada un contexto mínimo del sistema a consultar, para generar una expresión en un lenguaje formal que pueda ajustarse correctamente al formato de la base de conocimiento objetivo y obtener respuestas correctas sobre la misma.

En ese sentido la interrogante científica que compete a esta investigación es: ¿Qué tan efectivos pueden ser los LLMs para extraer información de bases de conocimientos a partir

de una descripción de la estructura de esta y sin necesidad de ser entrenado con datos específicos o poseer en su entrada ejemplos de cómo realizar la tarea a llevar a cabo?

Finalmente, para abordar dicha interrogante se tiene como hipótesis que, utilizando enfoques basados en el aprendizaje *zero-shot* con un LLM y teniendo como dato el esquema de una base de conocimiento a consultar, es posible obtener mejores resultados que utilizando directamente el LLM con la consulta a realizar por el usuario.

Objetivos

Dadas las ideas anteriores, los objetivos principales del trabajo consisten en diseñar e implementar una estrategia experimental capaz de verificar la capacidad de los LLMs para la consulta en lenguaje natural a bases de conocimiento estructuradas con independencia del dominio, para lo cual se empleará un enfoque basado en el aprendizaje *zero-shot*.

Relevancia del estudio

Este estudio aborda una problemática significativa en el campo de las bases de datos de grafos, particularmente, la complejidad y la accesibilidad limitada de los lenguajes de consulta como Cypher de Neo4J. A pesar de la potencia de estas bases de datos, su uso se ve restringido a individuos con conocimientos técnicos avanzados. Nuestra investigación introduce el uso del aprendizaje *zero-shot* en el procesamiento del lenguaje natural para traducir consultas en lenguaje natural a código Cypher, facilitando así su accesibilidad a un público más amplio. Los resultados experimentales muestran mejoras significativas en la precisión de las traducciones, lo cual no solo avanza en la técnica de procesamiento de lenguaje natural, sino que también tiene el potencial de democratizar el uso de bases de datos avanzadas, ampliando su aplicabilidad en diversas áreas académicas y sectores industriales. Esta contribución es especialmente relevante en la era de la información, donde la eficiencia y la accesibilidad de la tecnología de bases de datos tienen un impacto directo en la capacidad de gestión y análisis de grandes volúmenes de datos.

1. Propuesta de Solución

Dentro del ámbito de las bases de datos y las consultas que las acceden, existe una tarea particularmente compleja: traducir preguntas formuladas en lenguaje natural a un lenguaje de consulta estructurado como Cypher. Dada una pregunta Q , formulada en lenguaje cotidiano, y un esquema de base de datos S , el cual se compone a partir del tuplo N, A, R , donde encontramos múltiples nodos N (que representan instancias de una entidad en la base de datos), atributos A (tanto en nodos como en relaciones) y relaciones entre pares de nodos R . La problemática subyacente en el proceso de convertir *Text-to-Cypher* se centra en generar una consulta en lenguaje Cypher Y que sea equivalente y responda adecuadamente a la pregunta inicial Q realizada por un usuario humano.

1.1 LLM para resolver *Text-to-Cypher*

Con el auge de la inteligencia artificial y el aprendizaje automático, la tarea de convertir texto en lenguaje natural a código en Cypher ha sido recientemente abordada a través de técnicas modernas. En trabajos más recientes, algunos investigadores [18, 12] han abogado por formular esta tarea como un desafío de generación. Utilizando lo que se conoce como *prompts* o indicaciones P , es posible dirigir y guiar a un gran modelo de lenguaje M en esta labor. Este modelo, una vez entrenado, puede estimar una distribución de probabilidad sobre posibles consultas Cypher Y . De esta forma, el modelo es capaz de generar, paso a paso y *token por token*, una consulta apropiada.

La fórmula subyacente para generar la consulta Y se estructura como:

$$P_M(Y|P, S, Q) = \prod_{i=1}^{|Y|} P_M(Y_i|P, S, Q, Y < i).$$

Para simplificar, $Y < i$ se refiere al fragmento inicial o pre-fijo de la consulta Cypher que se está construyendo. Mientras que $P_M(Y_i|*)$ denota la probabilidad condicional asociada con la generación del i -ésimo *token*, considerando factores como el prefijo existente $Y < i$, la indicación P , el esquema S y la pregunta original Q .

Enfoques considerados

El desarrollo de un sistema con un LLM para hacer inferencias de *zero-shot* y generar Cypher implica varios enfoques y desafíos.

Primero, se exploraron modelos de código abierto. El primer modelo probado fue Alpaca Lora 7B [2]. Sin embargo, la calidad de las inferencias no fue la adecuada. En algunos casos, las respuestas eran Cypher no compilable con errores de sintaxis. En otros casos, el modelo generaba otro lenguaje de consulta que no era Cypher, como por ejemplo SQL.

Posteriormente, se probó con GPT4All [1] y con Vicuna 7B [19], obteniendo resultados similares. Estos modelos, aunque útiles, no proporcionaban la precisión y la generación de Cypher que se requería para el sistema.

2. Arquitectura general

En el desarrollo de sistemas de bases de datos, la interacción eficiente y la gestión de esquemas son fundamentales para la manipulación y el mantenimiento de datos.

En este contexto, se ha desarrollado una arquitectura de software compuesta por varios componentes interconectados diseñados para interactuar con una base de datos de grafos, con especial atención en Neo4J, un sistema de manejo de bases de datos basado en grafos.

Uno de los componentes clave de esta arquitectura es el GraphContractor, un módulo diseñado para facilitar la interacción con la base de datos. Este componente actúa como intermediario entre la aplicación y la base de datos,

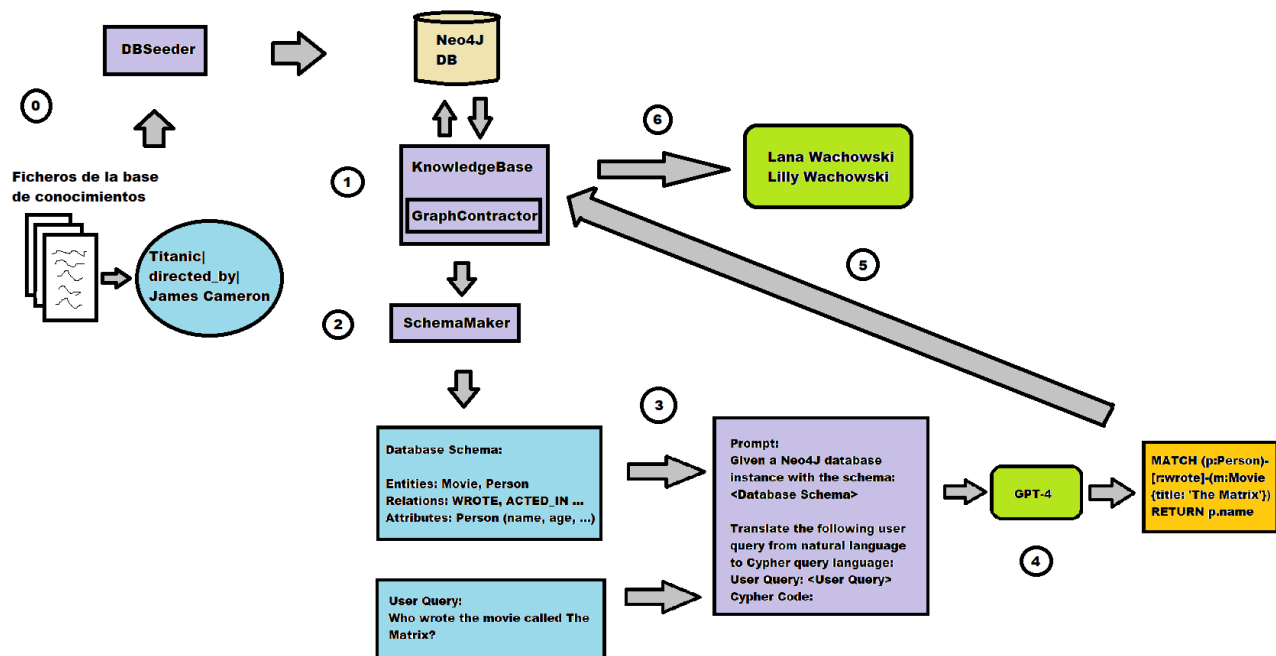


Figura 1. Arquitectura y funcionamiento de la propuesta de solución [Architecture and operation of the proposed solution].

manejando la lógica necesaria para establecer conexiones, ejecutar consultas y manejar los resultados.

La modularidad del *GraphContractor* permite su reutilización y fácil mantenimiento, además de proporcionar una capa de abstracción que simplifica las operaciones de la base de datos para los desarrolladores.

Para la generación de esquemas de la base de datos, se ha creado el *SchemaMaker*, una herramienta automatizada que se encarga de construir los esquemas necesarios para *Neo4J*. Esta herramienta juega un rol crucial en la estructuración de la base de datos, ya que define la organización de nodos, relaciones, propiedades y restricciones.

El *SchemaMaker* garantiza que la base de datos esté correctamente configurada para cumplir con los requisitos del dominio y las necesidades de la aplicación, asegurando así la integridad y coherencia de los datos.

En la interfaz entre el lenguaje natural y la base de datos, se ha integrado el modelo de lenguaje GPT-4, utilizado como una "caja negra" para la traducción de consultas en lenguaje natural a *Cypher*, el lenguaje de consulta para *Neo4J*. La capacidad de GPT-4 para comprender y generar texto hace posible que los usuarios realicen consultas complejas sin necesidad de conocer la sintaxis específica de *Cypher*. Para mejorar la precisión y relevancia de las traducciones, se ha elaborado una plantilla de *prompt* que se nutre de la salida del *SchemaMaker*. Esta plantilla guía al modelo de lenguaje proporcionando contexto y estructura, lo que permite que GPT-4 genere consultas *Cypher* más precisas y eficientes.

Finalmente, se ha desarrollado el *DBSeeder*, un componente encargado de poblar la base de datos de *Neo4J* con datos iniciales o de prueba. Utilizando consultas *Cypher* generadas

por el modelo de lenguaje GPT-4, el *DBSeeder* trabaja en conjunto con el *GraphContractor* para insertar nodos, atributos y relaciones en la base de datos. Esta funcionalidad es especialmente valiosa en las etapas de desarrollo y prueba, donde se requiere de una base de datos poblada para validar el diseño y la lógica de la aplicación. Además, el *DBSeeder* está diseñado para operar dentro de un contenedor de *Docker*, lo que ofrece ventajas significativas en términos de portabilidad, escalabilidad y aislamiento del entorno de desarrollo.

Cada uno de estos componentes representa un eslabón en la cadena de herramientas que permitió interactuar con la base de datos de grafos de manera más intuitiva y automatizada. La integración de modelos avanzados como GPT-4 en el proceso no solo mejora la accesibilidad para los usuarios finales sino que también, agiliza el ciclo de desarrollo, ofreciendo un enfoque moderno y eficiente en la gestión de bases de datos de grafos como *Neo4J*.

Para una mejor comprensión de la figura 1 se añadieron un conjunto de íncides que resaltan las distintas fases por las que pasa el sistema implementado. A continuación se enumeran y explican en qué consisten cada una de estas etapas:

1. **Paso 0:** Este representa el proceso mediante el cual se construye una instancia de una base de datos *Neo4J* a partir de un conjunto de ficheros de texto. Esta tarea se puede llevar a cabo mediante el componente *DBSeeder*, el cual puede programarse con funcionalidades específicas de acuerdo al formato de los ficheros iniciales y los datos que estos contienen. El componente *DBSeeder* se apoya del componente denominado *GraphContractor* internamente para realizar peticiones a la base de datos con el objetivo de añadir nuevos

registros de información, traduciendo la creación de entidades, relaciones y atributos en instrucciones de *Cypher* ejecutadas sobre una base de datos *Neo4J* objetivo.

2. **Paso 1:** En esta fase, el componente *KnowledgeBase* extrae la información referente a las entidades, relaciones y atributos de una base de datos *Neo4J* existente. Para ello, hace uso internamente de una instancia de un *GraphContractor*.
3. **Paso 2:** En este paso, se utiliza el *SchemaMaker*, el cual recibe la información de la base de datos procedente del componente *KnowledgeBase* y produce una descripción verbalizada y legible en lenguaje natural sobre la estructura de la instancia de *Neo4J* objetivo.
4. **Paso 3:** En este momento del proceso general, se recibe una consulta descrita en lenguaje natural humano sobre una solicitud de datos de la base de datos objetivo. Luego esta es utilizada junto con el esquema de la base de datos obtenido del *SchemaMaker* para conformar un texto de entrada al modelo GPT-4 con las características mencionadas.
5. **Paso 4:** Una vez obtenido el texto de entrada para realizar una inferencia con el modelo GPT-4, se procede a hacer una ejecución de este, produciendo así como salida un texto que representa un código en lenguaje *Cypher*.
6. **Paso 5:** En esta etapa, se utiliza la salida del modelo GPT-4 para ser ejecutada sobre la base de datos *Neo4J* objetivo mediante la herramienta *KnowledgeBase*.
7. **Paso 6:** Finalmente, se obtiene la respuesta procedente de la base de datos *Neo4J* con la información solicitada.

Con lo anteriormente explicado se expone un ejemplo de caso de uso donde se tienen como entradas al sistema una instancia de una base de datos *Neo4J* y una consulta en lenguaje humano para obtener como salida final el conjunto de datos extraídos de dicha base de datos objetivo correspondientes a la solicitud textual dada sobre estos.

3. Análisis Experimental

En este capítulo se presentan los marcos experimentales utilizados para evaluar la efectividad del sistema propuesto para la traducción de una consulta en lenguaje natural al lenguaje de consulta formal *Cypher*. Cada enfoque utilizado consistió en el uso de un conjunto de tuplas que contenían en común una consulta en lenguaje natural de ejemplo a traducir hacia un segundo elemento correspondiente con un objetivo a medir en la traducción.

Todos los experimentos fueron ejecutados en un servidor privado virtual (VPS) con sistema operativo *Ubuntu-20.04*,

memoria *RAM* de 16Gb, una *CPU* AMD basada en la arquitectura *x86_64*, con 8 núcleos y una velocidad de 2649.998 MHz y con un ancho de banda de 16Mb/s para la comunicación con servicios como la *API* de *OpenAI*.

El sistema de evaluación fue desarrollado sobre el *benchmark MetaQA* [23], el cual constituye el principal conjunto de datos de evaluación para la tarea *Text-to-Cypher* vista en la sección. En este caso se utilizó la versión clásica, donde los pares de evaluación consistían en una consulta en lenguaje natural con su correspondiente respuesta en la base de datos.

3.1 Evaluación sobre el *benchmark MetaQA* [23]

MetaQA [23] es un conjunto de datos diseñado para la tarea de razonamiento de múltiples pasos (*multi-hop*) en respuesta a preguntas. Está compuesto por entidades, relaciones y preguntas en lenguaje natural relacionadas con películas. Cada nodo en el grafo de conocimientos representa una entidad (como una película, actor o director), y las aristas representan relaciones entre las entidades. El conjunto de datos también incluye preguntas a tres niveles de complejidad (*1-hop*, *2-hop* y *3-hop*), con cada nivel requiriendo razonamiento sobre un número creciente de aristas en la base de datos en forma de grafos analizada. En la Tabla 1 se muestra un ejemplo de la distribución de dicho conjunto de datos.

	1-hop	2-hop	3-hop
Train	96,106	118,980	114,196
Dev	9,992	14,872	14,274
Test	9,947	14,872	14,274

Tabla 1. Distribución de los conjuntos de datos del *benchmark MetaQA* [*Distribution of benchmark MetaQA datasets*].

En este estudio solo se utilizarán los datos referentes a los conjuntos de evaluación (*Test*) para cada uno de los grupos especificados, ya que el modelo empleado es un gran modelo de lenguaje mediante la técnica *zero-shot*, por lo que no es necesario hacer un proceso de entrenamiento al mismo para realizar la tarea en cuestión.

Las principales métricas de evaluación utilizadas fueron el número de consultas que al ser traducidas a *Cypher* y ser ejecutadas sobre la base de conocimiento daban una respuesta idéntica a la respuesta objetivo (*correct*), así como el porcentaje de dichas consultas acertadas (*correct%*) sobre el total de consulta (*n*), número de consultas compiladas con éxito y su porcentaje correspondiente (*compiled* y *compiled%* respectivamente), algunas relacionadas con los recursos consumidos para el experimento como el costo monetario (*cost* (USD)) y el tiempo de ejecución de la evaluación en segundos (*elapsed_seconds*).

También, se calculó eficacia del modelo con respecto a cada tipo de consulta en el conjunto de evaluación y medidas clásicas como la precisión, el recobrado y la medida *F1* para evaluar la calidad de la extracción de información. La métrica *correct%* fue la utilizada para comparar el resultado del

modelo empleado sobre otros resultados en el estado del arte [1]. Además, implícitamente, al evaluar la efectividad del modelo sobre las consultas de los conjuntos de evaluación de *1-hop*, *2-hop* y *3-hop*, se evalúa la eficacia del modelo sobre consultas que requieren de una, dos y hasta tres relaciones de conexión respectivamente para encontrar la respuesta a la consulta.

Para la preparación del conjunto de datos se insertaron los elementos correspondientes a la base de conocimientos en una instancia de *Neo4J* con ayuda del componente *DBSeeder* visto en la sección. Luego se tomaron los conjuntos de prueba (*Test*) para *1-hop*, *2-hop* y *3-hop* y para cada par de evaluación se ejecutó el procedimiento descrito en el listado 2.

3.2 Resultados

Los resultados obtenidos para cada métrica analizada para cada conjunto de evaluación se muestran en la siguiente figura:

En la Tabla 2 se muestran los resultados de eficacia del modelo GPT-4 para traducir consultas a *Cypher* tal que puedan ser utilizadas para extraer información de la base de datos objetivo. En este caso, una consulta generada por el sistema utilizado fue considerada eficaz si al ejecutar el código de *Cypher* sobre la base de datos correspondiente, dicho resultado coincide con los datos de respuesta esperados asociados a cada par de los conjuntos de evaluación. Para aquellas consultas cuyo código de *Cypher* correspondiente requería de la presencia de una relación específica entre dos entidades en cuestión se tuvo un relevante resultado del 76,53 % de acierto. Por otro lado, aquellas consultas que requerían de la generación de una consulta con dos y hasta tres relaciones tuvieron como resultados unos discretos 43,45 % y 31,03 % respectivamente, lo que nos indica la deficiencia de este modelo para responder expresiones en lenguaje natural complejas que requieran acceder a la información de más una relación entre dos entes de la base de datos en forma de grafo.

Es importante resaltar de los resultados anteriores la efectividad del sistema para generar consultas de *Cypher* compilables, donde para cada lote de evaluación se obtuvieron valores sobre el 92 %, lo que confirma que se tuvo éxito en la inmensa mayoría de las veces que el modelo trató de traducir la consulta inicial en lenguaje natural a solamente un texto conteniendo un código de *Cypher*. En los casos donde dicho suceso no fue posible, se debió principalmente a que el modelo generó un texto adicional describiendo la consulta, ofrecía más de una consulta o simplemente había un error sintáctico en el código.

Conclusiones

Después de aplicar GPT-4 para la traducción de consultas al lenguaje *Cypher* mediante aprendizaje *zero-shot*, este estudio presenta conclusiones relevantes tanto en términos de fortalezas como de deficiencias. Destaca la eficiencia del modelo en generar código *Cypher* compilable, con un destacable 92 % de éxito en las pruebas, lo que subraya su competencia en la creación de consultas sin errores sintácticos o semánticos. Esta alta precisión es crucial para su aplicación práctica

en bases de datos como *Neo4J*.

Sin embargo, el modelo exhibe limitaciones significativas al manejar consultas más complejas. Mientras que en consultas sencillas (*1-hop*) la precisión es del 76,53 %, en consultas más complejas (*2-hop* y *3-hop*) esta precisión disminuye drásticamente a 43,45 % y 31,03 %, respectivamente. Esto indica que aunque GPT-4 es eficaz en traducciones simples, su rendimiento se ve comprometido en escenarios que involucran múltiples relaciones entre entidades.

Además, se identificaron varias áreas críticas no abordadas en el estudio, como la incapacidad del modelo para manejar consultas anidadas y funciones de agregación. Esto limita su utilidad en análisis de datos más complejos. La falta de un análisis multidominio también plantea preguntas sobre la capacidad de generalización del modelo, un factor esencial para determinar su eficacia en diferentes contextos y bases de datos.

Otro aspecto a mejorar es el formato básico de las respuestas generadas, que no satisface necesidades de análisis de datos más detallado y avanzado. Además, se señala que el tamaño limitado del esquema de la base de datos utilizada no puso a prueba la capacidad del modelo para manejar esquemas más grandes, una limitación importante para su aplicación práctica.

A pesar de estos desafíos, el sistema de traducción propuesto supera al modelo GPT-3 en la traducción de lenguaje natural a *Cypher* en el *benchmark* *MetaQA*, aunque todavía no alcanza los mejores resultados en la extracción de conocimiento usando *Cypher*.

Gracias a las medidas de precisión, recobrado y medida *F1* utilizadas para evaluar la capacidad del modelo para la extracción de información, considerando los verdaderos positivos, falsos positivos y falsos negativos, demuestran que, a pesar de no obtener todos los resultados esperados en ciertas consultas, es importante entender la distancia entre la respuesta dada y la esperada.

Finalmente, es evidente que la eficacia del modelo disminuye con el aumento de la complejidad de las consultas, especialmente en aquellas que requieren la predicción correcta de las direcciones de relaciones entre un mayor número de entidades. Este estudio deja claro que, mientras que el uso de un Gran Modelo de Lenguaje con la técnica de aprendizaje *zero-shot* muestra una eficiencia notable en ciertos aspectos, aún hay un camino considerable por recorrer para mejorar su rendimiento en escenarios más complejos y variados.

Recomendaciones

Para futuros trabajos relacionados con la aplicación de grandes modelos de lenguajes en la traducción de consultas en lenguaje natural a *Cypher* para interactuar con una base de conocimientos, se sugieren algunas direcciones clave. Es crucial perfeccionar la comprensión de consultas complejas, incluyendo las que involucran múltiples relaciones y funciones de agregación, para mejorar la precisión y versatilidad del modelo. Además, ampliar el esquema de la base de datos y

	n	compiled	correct	compiled %	correct %
hop 1	9947	9386	7613	94.36	76.53
hop 2	14872	13733	6462	92.34	43.45
hop 3	14274	13221	4430	92.62	31.03

Tabla 2. Resultados de ejecutar GPT-4 en los conjuntos de datos de prueba de *MetaQA* para *hop1*, *hop2* y *hop3* [Results of running GPT-4 on the *MetaQA* test data sets for *hop1*, *hop2* and *hop3*].

Método	1-hop	2-hop	3-hop
SOTA	97.50	98.80	94.80
zero-shot	24.75	6.37	9.72
zero-shot-cot	18.41	12.86	21.89
zero-shot+graph	91.69	46.82	19.40
zero-shot-cot+graph	86.16	47.36	19.29
zero-shot+graph+change-order	95.20	40.48	20.17
zero-shot-cot+graph+change-order	95.87	47.71	23.95
zero-shot Cypher Generation	30.00	10.00	13.00
GPT-4 zero-shot Cypher Generation	76.53	43.45	31.03
one-shot Cypher Generation	99.00	77.00	96.00

Tabla 3. Comparación de los resultados de otros modelos respecto al *benchmark MetaQA* [Comparison of the results of other models with respect to the *benchmark MetaQA*].

realizar pruebas en múltiples dominios podría proporcionar una evaluación más robusta de la capacidad de generalización del modelo. Finalmente, se recomienda mejorar el análisis y manejo de las estadísticas relacionadas con las respuestas incorrectas para afinar la precisión de las traducciones generadas.

Agradecimientos

Quisiera expresar mi sincero agradecimiento a NataSquad por su invaluable apoyo en el desarrollo de este proyecto. Su contribución no solo fue fundamental en términos financieros, sino que también facilitaron el acceso a recursos de cómputo esenciales que fueron cruciales para la realización de los experimentos. Además, agradezco especialmente por proveer una clave de la API de OpenAI, lo cual me permitió realizar pruebas efectivas con el modelo GPT-4, enriqueciendo significativamente los resultados de nuestra investigación. Su apoyo ha sido esencial para alcanzar los objetivos planteados y avanzar en este campo de estudio.

Suplementos

Este artículo no contiene información suplementaria.

Conflictos de interés

Declaro que he recibido financiación de NataSquad para la realización de este trabajo. Esta financiación incluyó el acceso a recursos de cómputo necesarios y una clave de la API de OpenAI para realizar pruebas con el modelo GPT-4. Aunque este apoyo ha sido esencial para el desarrollo y ejecución del proyecto, he tomado todas las medidas necesarias para asegurar que los resultados presentados sean el reflejo objetivo

de mis investigaciones, independientemente de los intereses comerciales de la empresa financiadora.

Referencias

- [1] AI, Nomic: *GPT4All*. <https://nomic-ai.com/>, 2023. <https://nomic-ai.com/>, Accedido el: 12/13/2023.
- [2] Alpaca: *Alpaca-Lora*. <https://alpaca.com/>, 2023. <https://alpaca.com/>, Accedido el: 12/13/2023.
- [3] Archive, Web: *Creación de una base de conocimiento*. <https://lc.cx/T9w20p>, 2023. <https://lc.cx/T9w20p>.
- [4] Bazaga, A., N. Gunwant y G. Micklem: *Translating synthetic natural language to database queries with a polyglot deep learning framework*. Scientific Reports, 11:18462, 2021. <https://doi.org/10.1038/s41598-021-98019-3>.
- [5] Bui, N. D. Q., H. Le, Y. Wang, J. Li, A. D. Gotmare y S. C. H. Hoi: *CodeTF: One-stop Transformer Library for State-of-the-art Code LLM*, 2023. <https://doi.org/10.48550/arXiv.2306.00029>, Accedido el: 05 de diciembre, 2023.
- [6] Deng, N., Y. Chen y Y. Zhang: *Recent Advances in Text-to-SQL: A Survey of What We Have and What We Expect*. En *Proceedings of the 29th International Conference on Computational Linguistics*, páginas 2166–2187, Online, October 2022. <https://aclanthology.org/2022.coling-1.190.pdf>, Presented at the 29th

- International Conference on Computational Linguistics, October 12-17, 2022.
- [7] GenAI, Meta: *Llama 2: Open Foundation and Fine-Tuned Chat Models*. Informe técnico, Simons Foundation, member institutions, 2023. <https://arxiv.org/abs/2307.09288>.
- [8] Huang, L., W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin y T. Liu: *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*. arXiv preprint arXiv:2311.05232, 2023. <https://arxiv.org/abs/2311.05232>.
- [9] Humza, N., U. K. Asad, Q. Shi, S. Muhammad, A. Saeed, U. Muhammad, A. Naveed, B. Nick y M. Ajmal: *A Comprehensive Overview of Large Language Models*. ArXiv, 2023. <https://arxiv.org/pdf/2307.06435.pdf>.
- [10] insights, SAP: *¿Qué es el ¿modelado de datos?* <https://www.sap.com/latinamerica/products/technology-platform/datasphere/what-is-data-modeling.html>, 2023. <https://www.sap.com/latinamerica/products/technology-platform/datasphere/what-is-data-modeling.html>.
- [11] Li, P., T. Sun, Q. Tang, H. Yan, Y. Wu, X. Huang y X. Qiu: *Large Code Generation Models are Better Few-Shot Information Extractors*. En *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, volumen 1, páginas 15339–15353, Association for Computational Linguistics, 2023. <https://aclanthology.org/2023.acl-long.855.pdf>, Accedido el: 05 de diciembre, 2023.
- [12] Liu, A., X. Hu, L. Wen y P. S. Yu: *A Comprehensive Evaluation of Chat-GPT's Zero-Shot Text-to-SQL Capability*. ArXiv, abs/2303.13547, 2023. <https://arxiv.org/abs/2303.13547>.
- [13] MAPFRE, Fundación: *¿Cuánta información se genera y almacena en el mundo?* <https://lc.cx/lpCjng>, 2023. <https://lc.cx/lpCjng>.
- [14] OpenAI: *GPT-4 Technical Report*. Informe técnico, Simons Foundation, member institutions, 2023. <https://arxiv.org/abs/2303.08774>.
- [15] Parkhi, O. M., S. M. Ali, M. Elgammal y C. K. I. Williams: *Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly*. ArXiv, 2017. <https://arxiv.org/pdf/1707.00600.pdf>.
- [16] Site, Neo4J Official: *Neo4J Graph Database: The most trusted database for intelligent applications*. <https://neo4j.com/product/neo4j-graph-database/>, 2023. <https://neo4j.com/product/neo4j-graph-database/>.
- [17] Site, Neo4J Official: *Query a Neo4J Database using Cypher*. <https://neo4j.com/docs/getting-started/cypher-intro/>, 2023. <https://neo4j.com/docs/getting-started/cypher-intro/>.
- [18] Sun, R., S. A. Arik, H. Nakhost, H. Dai, R. Sinha, P. Yin y T. Pfister: *SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL*. ArXiv, abs/2306.00739, 2023. <https://arxiv.org/abs/2306.00739>.
- [19] Vicuna: *Vicuna 7b*. <https://vicuna.com/>, 2023. <https://vicuna.com/>, Accedido el: 12/13/2023.
- [20] Wayne, X. Z., Z. Kun, L. Junyi, T. Tianyi, W. Xiaolei, H. Yupeng, M. Yingqian, Z. Beichen, Z. Junjie, D. Zican, D. Yifan, Y. Chen, C. Yushuo, C. Zhipeng, J. Jinhao, R. Ruiyang, L. Yifan, T. Xinyu, L. Zikang, L. Peiyu, N. Jian-Yun y W. Ji-Rong: *A Survey of Large Language Models*. ArXiv, 2023. <https://arxiv.org/pdf/2307.06435.pdf>.
- [21] Website, IBM Official: *Lenguaje de consulta estructurada (SQL)*. <https://n9.cl/lenguajeconsulta>, 2023. <https://n9.cl/lenguajeconsulta>.
- [22] Wikipedia: *Graph database*. <https://n9.cl/n7atf>, 2023. <https://n9.cl/n7atf>.
- [23] Yuyu, Z., D. Hanjun, K. Zornitsa, J. S. Alexander y S. Le: *Variational Reasoning for Question Answering with Knowledge Graph*. En *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017. <https://arxiv.org/abs/1709.04071>.

