

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“EVALUACIÓN DE MODELO DE PREDICCIÓN DE LATENCIA EN CONSULTAS SPARQL”

AXEL DILAN ARROYO GÓMEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Carlos Buil Aranda
Profesor Correferente: Carlos Valle Vidal

Agosto - 2024

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a todas aquellas personas que me han acompañado y apoyado a lo largo de este camino. En primer lugar, a mi familia, pilar fundamental en mi vida. A mis padres y madrina, por su amor incondicional y constante apoyo. A mis hermanas, por su compañía, sus palabras de aliento y por creer siempre en mí. A mis amigos, por ser mi soporte emocional, por las risas compartidas que aliviaron los momentos de estrés y que hicieron el pasar de estos años una experiencia grata. Un agradecimiento especial a mi profesor guía, Carlos Buil, por su invaluable orientación, paciencia y dedicación, indispensable para sacar este trabajo adelante.

RESUMEN

Resumen— El uso de bases de datos orientadas a grafos ha crecido significativamente debido a su capacidad para manejar grandes volúmenes de datos y sus complejas interconexiones. La Web Semántica, respaldada por estándares como RDF y SPARQL, permite representar y consultar datos de manera eficiente, facilitando la interoperabilidad y el intercambio de información en la web. En este contexto, la predicción de la latencia de consultas SPARQL es crucial para optimizar el rendimiento de los sistemas basados en SPARQL, reduciendo tiempos de respuesta y mejorando la experiencia del usuario. Utilizando el dataset LSQ 2.0 sobre consultas del mundo real en DBpedia se evalúa un modelo de predicción, demostrando su eficacia y proporcionando una base para futuras mejoras en la optimización de consultas SPARQL.

Palabras Clave— Web Semántica; SPARQL; Bases de Datos de Grafos; Predicción de Latencia; Aprendizaje Profundo

ABSTRACT

Abstract— The use of graph-oriented databases has significantly grown due to their ability to handle large volumes of data and complex interconnections. The Semantic Web, supported by standards such as RDF and SPARQL, enables efficient data representation and querying, facilitating interoperability and information exchange on the web. In this context, predicting SPARQL query latency is crucial for optimizing the performance of SPARQL-based systems, reducing response times, and enhancing user experience. By utilizing the LSQ 2.0 dataset on real-world queries in DBpedia, a prediction model is evaluated, demonstrating its effectiveness and providing a foundation for future improvements in SPARQL query optimization.

Keywords— Semantic Web; SPARQL; Graph Databases; Latency Prediction; Deep Learning.

GLOSARIO

BGP: Basic Graph Pattern

DB: Database / Base de datos

GDB: Graph Database / Base de datos de grafos

IRI: Internationalized Resource Identifier

KEGG: Kyoto Encyclopedia of Genes and Genomes

LOD: Linked Open Data

LSQ: Linked SPARQL Queries

MAE: Mean Absolute Error

MAPE: Mean Absolute Percentage Error

ML: Machine Learning

MSE: Mean Squared Error

NoSQL: Not Only SQL

OWL: Web Ontology Language

PCA: Principal Component Analysis

QEP: Query Execution Plan

QoS: Quality of Service

RDBMS: Relational Database Management System / Sistema de gestión de bases de datos relacionales

RDF: Resource Description Framework

RDFS: RDF Schema

RMSE: Root Mean Squared Error

SKOS: Simple Knowledge Organization System

SPARQL: SPARQL Protocol and RDF Query Language

SQL: Structured Query Language

TBCNN: Tree-Based Convolutional Neural Network

t-SNE: t-distributed Stochastic Neighbor Embedding

URI: Uniform Resource Identifier

W3C: World Wide Web Consortium

XML: eXtensible Markup Language

ÍNDICE DE CONTENIDOS

RESUMEN	III
ABSTRACT	III
GLOSARIO	IV
ÍNDICE DE FIGURAS	VII
ÍNDICE DE TABLAS	VIII
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Contexto	3
1.2 Problema	3
1.3 Objetivos	4
1.3.1 Objetivo General	4
1.3.2 Objetivos específicos	4
1.4 Alcance de la solución	4
CAPÍTULO 2: MARCO CONCEPTUAL	5
2.1 Base de datos de grafo	5
2.2 Web Semántica	6
2.2.1 Importancia de la Web Semántica	7
2.3 RDF (Resource Description Framework)	8
2.3.1 Fundamentos de RDF	9
2.3.2 Serialización de RDF	9
2.3.3 Vocabularios RDF y RDF Schema	10
2.3.4 Ejemplo de RDF	10
2.4 SPARQL	11
2.5 Sistemas de Bases de Datos y Planes de Ejecución	13
2.5.1 Planificación y Ejecución de Consultas	14
2.6 Algoritmos de Aprendizaje Automático	15
2.6.1 Aprendizaje Supervisado	15
2.6.2 Aprendizaje No Supervisado	15
2.6.3 Hiperparámetros	16
2.6.4 Conjuntos de Entrenamiento, Validación y Prueba	16
2.6.5 Métricas de Evaluación	16
2.7 Predicción de latencia	17
2.7.1 Predicción de latencia en RDBMS	17
2.7.2 Modelos de Predicción en SPARQL	18
2.7.3 Codificación de consultas	18

2.8 LSQ 2.0: Un conjunto de datos de consultas SPARQL	19
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	21
3.1 Modelo de predicción	21
3.2 Extracción de consultas y características	22
3.3 Exploración de datos	25
3.3.1 Distribución de tiempo	25
3.3.2 Correlaciones	25
3.3.3 Información Mutua	27
3.4 Experimentos	30
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	33
4.1 Resultados y evaluación	33
4.1.1 DBPedia	33
4.1.2 LinkedGeoData	35
4.1.3 DBPedia + LinkedGeoData + KEGG	37
4.2 Generalización entre datasets	39
CAPÍTULO 5: CONCLUSIONES	40
5.1 Trabajo futuro	42
ANEXOS	44
REFERENCIAS BIBLIOGRÁFICAS	49

ÍNDICE DE FIGURAS

1	Ejemplo de grafo.	6
2	Un grafo de RDF con dos nodos (sujeto y objeto) y un arco que los conecta (predicado).	9
3	Tripletas RDF (informales)	11
4	Grafo de las tripletas de la figura 3	11
5	Un grafo RDF que guarda información acerca de servicios de transporte entre ciudades.	13
6	Fases del procesamiento de consultas	14
7	Modelo de datos de LSQ	20
8	Arquitectura genérica de la red neuronal, las unidades por capas son válidas a modo de ejemplo	21
9	Distribución de datos de conjunto de entrenamiento de dataset DBPedia	25
10	Distribución de datos de conjunto de entrenamiento de dataset DBPedia, filtradas las consultas que no devuelven resultados. Fuente: Elaboración Propia.	26
11	Distribución de datos de conjunto de entrenamiento de dataset LinkedGeoData. Fuente: Elaboración Propia.	26
12	Distribución de datos de conjunto de entrenamiento de dataset combinado: DBPedia + LinkedGeoData + KEGG. Fuente: Elaboración Propia	27
13	Correlación entre variables del dataset DBPedia	28
14	Diagrama de dispersión entre projectVarCount y tiempo de ejecución del dataset DBPedia	29
15	Información mutua entre variables del dataset DBPedia	30
16	Distribución de datos de conjunto de entrenamiento de dataset combinado: DBPedia + LinkedGeoData + KEGG luego de realizar over-sampling con 10 bins y un límite de 60 %	32
17	Entrenamiento en DBPedia	34
18	Predicciones en DBPedia	35

19	Entrenamiento en DBPedia con NuSVR	36
20	Predicciones en DBPedia con NuSVR	37
21	Entrenamiento en LinkedGeoData	38
22	Predicciones en LinkedGeoData para dataset regular	39
23	Entrenamiento en dataset combinado: DBPedia + LinkedGeoData + KEGG . . .	40
24	Predicciones en dataset combinado: DBPedia + LinkedGeoData + KEGG	41

ÍNDICE DE TABLAS

1	Cantidad de consultas obtenidas	23
2	Cantidad de consultas luego de realizar over-sampling a dataset DBPedia + LinkedGeoData + KEGG	31
3	Porcentaje de predicciones según color/categoría en DBPedia	35
4	Porcentaje de predicciones según color/categoría en DBPedia con NuSVR . . .	35
5	Porcentaje de predicciones según color/categoría en LinkedGeoData para dataset regular	37
6	Porcentaje de predicciones según color/categoría en LinkedGeoData para dataset aumentado con 10 bins al 40 %	37
7	Métricas de error de predicciones en conjunto de test para dataset combinado	38
8	Porcentaje de predicciones mejor ejecución según color/categoría en dataset combinado: DBPedia + LinkedGeoData + KEGG	39
9	Métricas de evaluación de resultados en conjunto de test para DBPedia en distintos datasets y variaciones del modelo con nuevas características LSQ, los valores presentan Promedio $\pm$ Desviación Estándar de 100 ejecuciones . .	44
10	Métricas de evaluación de resultados en conjunto de test para DBPedia en distintos datasets usando modelo NuSVR, los valores presentan Promedio $\pm$ Desviación Estándar de 100 ejecuciones	44
11	Métricas de evaluación de resultados en conjunto de test para LinkedGeoData en distintos datasets, los valores presentan Promedio $\pm$ Desviación Estándar de 100 ejecuciones	45

12	RMSE en conjunto de test para DBPedia, resultados para 100 ejecuciones . . .	45
13	MAE en conjunto de test para DBPedia, resultados para 100 ejecuciones . . .	46
14	R2 en conjunto de test para DBPedia, resultados para 100 ejecuciones . . .	46
15	MAPE en conjunto de test para DBPedia, resultados para 100 ejecuciones . . .	47
16	RMSE en conjunto de test para LinkedGeoData	47
17	MAE en conjunto de test para LinkedGeoData	48
18	R2 en conjunto de test para LinkedGeoData	48
19	MAPE en conjunto de test para LinkedGeoData	48

INTRODUCCIÓN

El área de las bases de datos ha evolucionado significativamente para abordar las distintas necesidades surgidas con el tiempo. Esto ha llevado a la aparición de sistemas NoSQL, que priorizan el procesamiento de grandes volúmenes de datos sobre la coherencia de la información, diferenciándose así de las tradicionales bases de datos relacionales (SQL).

Entre los sistemas NoSQL se tienen las bases de datos orientadas a grafos, cuyo valor radica en almacenar información estructurada en función de sus relaciones. Este enfoque ha ganado relevancia en los últimos años, impulsado por aplicaciones como el análisis de interacciones en redes sociales o la representación de redes de transporte.

La web semántica, una extensión de la web actual, busca mejorar la interoperabilidad de la información al permitir a las máquinas entender y responder a solicitudes humanas complejas basadas en el significado de los datos. Tecnologías como RDF (Resource Description Framework) y SPARQL (SPARQL Protocol and RDF Query Language) son fundamentales en este contexto. RDF permite representar la información en forma de tripletas (sujeto-predicado-objeto), mientras que SPARQL es el lenguaje estándar para consultar estas bases de datos RDF, permitiendo realizar consultas complejas sobre grafos de datos.

Este auge ha generado una demanda creciente de motores de bases de datos que soporten consultas navegacionales para explorar conexiones entre datos o nodos. Sin embargo, las bases de datos de grafos, como las basadas en el estándar RDF y que utilizan SPARQL como lenguaje de consulta, enfrentan desafíos particulares. Entre estos, destaca la necesidad de optimizar el tiempo de ejecución de las consultas, dado que las bases de datos pueden almacenar terabytes de información, haciendo que la latencia sea un factor crítico.

SPARQL (SPARQL Protocol and RDF Query Language), el lenguaje estándar para consultar bases de datos RDF permite realizar consultas complejas sobre grafos de datos, evaluando patrones de grafo para obtener la información deseada. La ejecución eficiente de estas consultas depende del plan de ejecución (QEP, por sus siglas en inglés) generado por el motor de optimización de la base de datos. Dado que existen miles de posibles planes de ejecución, los motores suelen conformarse con encontrar un plan suficientemente bueno.

La predicción precisa de la latencia de una consulta es crucial para optimizar los planes de ejecución y mejorar la asignación de recursos, especialmente en entornos complejos como los servicios de bases de datos en la nube. Según [Pavlo *et al.*, 2017] el tiempo de ejecución de una consulta es la métrica más importante en cualquier motor de base de datos, pues captura todos los aspectos del rendimiento. Estimar la latencia de una consulta no solo ayuda en la optimización del plan de consulta, sino que también es esencial para cumplir con objetivos de calidad de servicio (QoS) y mejorar la experiencia del usuario [Akdere *et al.*, 2012, Duggan *et al.*, 2011].

La falta de conocimiento preciso sobre la latencia de las consultas SPARQL dificulta la opti-

mización de estas. Por ello, este trabajo se enfoca en evaluar un modelo de predicción de latencia utilizando técnicas de Deep Learning, validándolo con un conjunto de datos más amplio y variado, como el dataset LSQ 2.0. Este enfoque no solo permitirá mejorar la predicción de latencia, sino que también abrirá nuevas oportunidades para la optimización de consultas SPARQL.

El presente trabajo se organiza de la siguiente forma:

- **Capítulo 1 - Definición del problema:** Se contextualiza y se identifica el problema a tratar en mayor profundidad, se define el objetivo general y objetivos específicos, y el alcance de la solución propuesta.
- **Capítulo 2 - Marco Conceptual:** Se abordan diferentes conceptos relevantes en el contexto del trabajo.
- **Capítulo 3 - Propuesta de solución:** Se describe el diseño de la solución propuesta para resolver el problema identificado.
- **Capítulo 4 - Validación de la solución:** Se presentan los resultados que validan la solución presentada y el cumplimiento de los objetivos expuestos.
- **Capítulo 5 - Conclusiones:** Se presentan las conclusiones referentes al tema estudiado y solución propuesta, además se definen futuras extensiones del trabajo desarrollado.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. Contexto

El avance de las tecnologías de la información ha permitido la proliferación de grandes volúmenes de datos estructurados y no estructurados, entre estos últimos, los datos en formato RDF (Resource Description Framework) y las consultas SPARQL (SPARQL Protocol and RDF Query Language) se destacan por su capacidad para representar y consultar información altamente interrelacionada. Las bases de datos de grafos, como las utilizadas en la web semántica, permiten el almacenamiento y procesamiento eficiente de estas estructuras complejas, facilitando la extracción de conocimientos valiosos a partir de grandes conjuntos de datos.

SPARQL es un lenguaje de consulta estandarizado para datos RDF, que permite formular preguntas complejas sobre datos interconectados. Sin embargo, la latencia de las consultas SPARQL puede variar significativamente dependiendo del tamaño y la estructura de los datos, así como del plan de ejecución seleccionado por el motor de consulta. Predecir la latencia de las consultas es crucial para optimizar el rendimiento del sistema y mejorar la experiencia del usuario.

1.2. Problema

La predicción de la latencia de consultas SPARQL es un desafío complejo debido a la variedad de factores que influyen en el tiempo de ejecución. Estos factores incluyen la estructura de la consulta, el tamaño del conjunto de datos, la distribución de los datos, y el plan de ejecución utilizado por el motor de consulta. En [Casals *et al.*, 2023] se presenta un modelo de predicción de latencia usando Deep Learning, basado en las consultas realizadas en el endpoint de SPARQL de Wikidata 2018, ejecutadas en una versión local de Wikidata dumps 2019 ha demostrado la viabilidad de usar modelos de aprendizaje profundo para esta tarea. Sin embargo, este enfoque tiene limitaciones en cuanto a la generalización y la aplicabilidad a otros conjuntos de datos y motores de consulta.

El presente trabajo se propone validar un modelo de predicción de latencia de consultas SPARQL utilizando un conjunto de datos distinto y variado, LSQ 2.0, que incluye ejecuciones y consultas realizadas en el mundo real. Este enfoque tiene como objetivo proporcionar una evaluación más robusta y fiable del modelo, contribuyendo a la optimización de consultas en diversos contextos y mejorando la eficiencia de los sistemas de bases de datos RDF.

1.3. Objetivos

1.3.1. Objetivo General

Validar un modelo de predicción de latencia de consultas SPARQL utilizando un conjunto de datos LSQ 2.0, comparando su rendimiento con el trabajo original basado en los logs de Wikidata 2018 y consultas ejecutadas en una versión local de Wikidata dumps 2019.

1.3.2. Objetivos específicos

1. **Preparar el conjunto de datos:** Identificar y extraer la información relevante del dataset LSQ, realizar un análisis exploratorio de los datos y hacer las transformaciones necesarias para ser compatible con el modelo, seleccionando las características más relevantes.
2. **Implementar el modelo de predicción:** Aplicar el modelo existente al conjunto de datos, ajustando los parámetros y métodos según sea necesario.
3. **Evaluar el rendimiento del modelo:** Comparar los resultados obtenidos contra los resultados del trabajo original, utilizando métricas de desempeño adecuadas.
4. **Analizar los resultados:** Interpretar los hallazgos y discutir las implicaciones para la optimización de consultas SPARQL en diferentes contextos.

1.4. Alcance de la solución

El alcance de esta investigación se centra en la validación de un modelo de predicción de latencia de consultas SPARQL utilizando un conjunto de consultas más variado y realista, LSQ 2.0. Esto incluye la preparación y procesamiento de los datos, la implementación y ajuste del modelo de predicción, y la evaluación de su rendimiento. No se abordarán en profundidad temas relacionados con la optimización de consultas o la implementación de nuevos algoritmos de consulta, aunque se discutirán brevemente en el contexto de los resultados obtenidos. Esta investigación contribuirá a la comprensión de la aplicabilidad y eficacia de modelos de predicción de latencia en entornos de datos RDF y consultas SPARQL, proporcionando una base para futuras mejoras en la optimización de sistemas de bases de datos semánticas.

CAPÍTULO 2

MARCO CONCEPTUAL

Este capítulo presenta los conceptos fundamentales necesarios para comprender el trabajo realizado. Se abordarán las tecnologías de la Web Semántica, los algoritmos de aprendizaje automático y las técnicas de predicción de rendimiento en consultas SPARQL. Estos conceptos son esenciales para entender el contexto y los métodos utilizados en la investigación sobre la predicción de latencia en consultas SPARQL mediante redes neuronales profundas

2.1. Base de datos de grafo

Un grafo es una estructura compuesta por nodos o vértices unidos entre sí por aristas o arcos, lo que permite representar relaciones entre los diversos elementos que lo componen. Las definiciones formales de un grafo varían según el modelo que se esté usando, como los grafos etiquetados por aristas *Edge-labelled graph*, grafos de propiedades *Property graph*, y grafos de dominio *Domain graph*. A continuación se omiten las definiciones formales de cada uno para enforarlos en su aplicación práctica.

En la figura 1 se muestra un pequeño ejemplo de un grafo, en donde se tienen dos tipos de nodos: Persona (*Person*) y Auto (*Car*). Cada uno posee atributos específicos que lo definen, como el nombre, fecha de nacimiento y cuenta de Twitter para las personas, y la marca y el modelo para los autos. Además, existen arcos que relacionan estos nodos, representando relaciones como *loves* entre personas, u *owns* entre una persona (Ann) y un auto. Es importante notar que las relaciones en este grafo son direccionales, lo que implica que, si se desea hacer que una relación sea bidireccional, es necesario declararla en ambos sentidos, como se observa en la relación *loves*.

Este sencillo ejemplo permite realizar consultas complejas, como obtener las personas que son dueñas de un auto de una marca específica, o las personas que viven con otra persona. La capacidad de representar y consultar estas relaciones de manera eficiente es lo que hace a las bases de datos de grafos especialmente útiles para aplicaciones que requieren modelar y analizar relaciones complejas, como redes sociales, sistemas de recomendación, o análisis de redes de transporte.

Las bases de datos de grafos se destacan por su capacidad para manejar estructuras de datos altamente conectadas, permitiendo ejecutar consultas transversales y de patrones con una gran eficiencia. Esto es particularmente ventajoso en comparación con las bases de datos relacionales tradicionales, donde la ejecución de consultas complejas que involucran múltiples uniones (*joins*) puede resultar en tiempos de respuesta significativamente más largos.

En resumen, las bases de datos de grafos ofrecen una solución poderosa y flexible para el al-

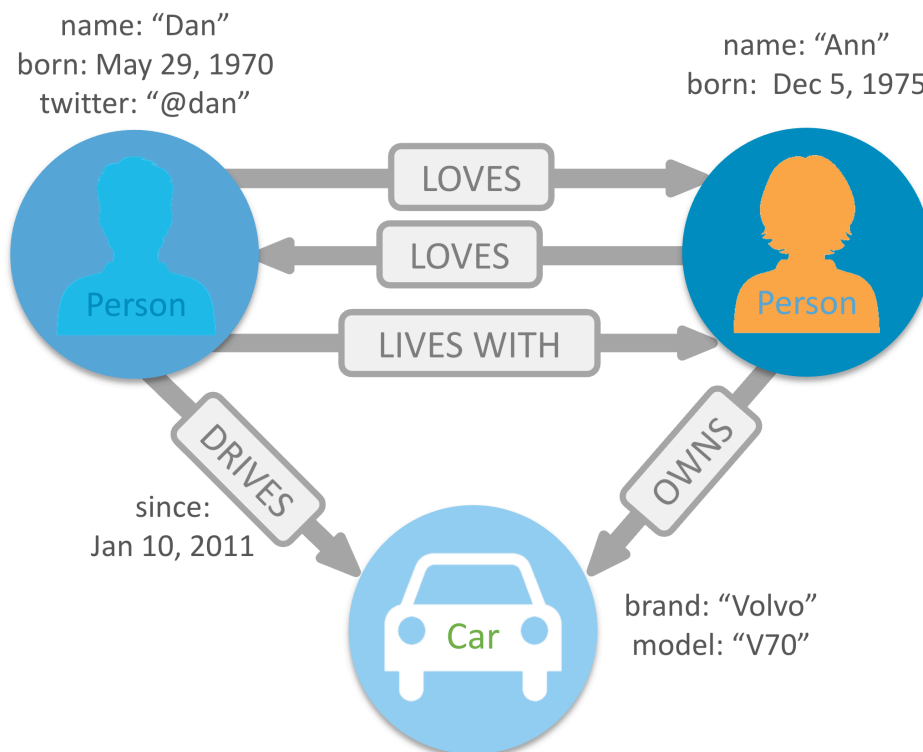


Figura 1: Ejemplo de grafo.

macenamiento y análisis de datos con relaciones complejas, facilitando la creación de aplicaciones avanzadas que dependen de la interconectividad de la información. Esto es especialmente relevante en contextos donde la estructura de las relaciones y las conexiones directas entre entidades son cruciales para el rendimiento y la eficiencia de las consultas.

2.2. Web Semántica

La Web Semántica es una extensión de la World Wide Web actual, donde la información está definida y enlazada de manera que pueda ser entendida y procesada tanto por personas como por máquinas. El principal objetivo de la Web Semántica, según el W3C (World Wide Web Consortium), es convertir la web en un espacio donde los datos puedan ser compartidos y reutilizados a través de diversas aplicaciones, empresas y comunidades.

El W3C define la Web Semántica como "una web de datos" que permite a los usuarios crear datos almacenados en la web, construir vocabularios y escribir reglas para manejar esos datos. Los objetivos principales incluyen:

1. **Interoperabilidad de Datos:** Facilitar la integración y combinación de datos de diferentes fuentes.

2. **Mejora de la Precisión en las Búsquedas:** Permitir que las búsquedas en la web sean más precisas al entender el contexto y el significado de la información.
3. **Automatización:** Habilitar la automatización de procesos que actualmente requieren intervención humana.

En la Web Semántica, los datos son tratados como “ciudadanos de primera clase”, lo que significa que los datos y los enlaces entre ellos son tan importantes como los documentos en los que están contenidos. Esto permite que los datos sean consultados, combinados y reutilizados de manera más efectiva, abriendo nuevas posibilidades para la inteligencia artificial y el análisis de datos.

Para cumplir con estos objetivos, el W3C ha desarrollado una serie de estándares y tecnologías que permiten la representación de datos de manera que sean fácilmente accesibles y comprensibles tanto por humanos como por máquinas, entre los cuales se incluyen:

- **XML (eXtensible Markup Language):** Un lenguaje de marcado que define una estructura de datos.
- **XML Schema:** Un lenguaje que describe la estructura y el contenido de los documentos XML.
- **RDF (Resource Description Framework):** Un marco para representar información sobre recursos en la web.
- **RDF Schema:** Un vocabulario para describir propiedades y clases de recursos RDF.
- **OWL (Web Ontology Language):** Un lenguaje más complejo para definir y instanciar ontologías en la web.
- **SPARQL (SPARQL Protocol and RDF Query Language):** Un lenguaje de consulta y protocolo para consultar datos almacenados en RDF.

2.2.1. Importancia de la Web Semántica

La generación de datos enlazados o *Linked Data* es uno de los componentes más cruciales de la Web Semántica. Linked Data se refiere a la práctica de estructurar y enlazar datos de tal manera que puedan ser fácilmente compartidos y conectados con otros datos en la web [Berners-Lee,]. Este enfoque promueve la interoperabilidad de datos, ya que facilita la integración de información de diferentes fuentes, proporcionando un contexto más rico y significativo.

El impacto de Linked Data en la interoperabilidad de datos es significativo. Al utilizar estándares como RDF y SPARQL, Linked Data permite que los datos sean accesibles y utilizables

por diferentes aplicaciones y sistemas sin la necesidad de conversiones complejas o pérdida de información. Esto ha revolucionado áreas como la investigación académica, la gestión de datos gubernamentales, y el desarrollo de aplicaciones web y móviles, permitiendo una colaboración más estrecha y un intercambio de información más eficiente.

Existen varios proyectos y aplicaciones notables que han surgido gracias a la Web Semántica y la generación de Linked Data:

- **Schema.org:** Un esfuerzo colaborativo iniciado por Google, Microsoft, Yahoo y Yandex, que proporciona un conjunto de esquemas para estructurar datos en la web. Esto facilita a los motores de búsqueda entender y procesar mejor la información en los sitios web, mejorando la visibilidad y accesibilidad de los datos.
- **Linked Open Data (LOD) Cloud:** La LOD Cloud es una colección de conjuntos de datos interconectados disponibles públicamente en la web. Estos datos cubren una amplia gama de dominios, incluyendo geografía, medios de comunicación, publicaciones académicas y más. La LOD Cloud ejemplifica cómo Linked Data puede ser utilizado para conectar y enriquecer información de múltiples fuentes, promoviendo la interoperabilidad y el descubrimiento de nuevos conocimientos.

La Web Semántica y la generación de Linked Data han transformado la manera en que los datos son compartidos y reutilizados en la web. A través de estándares y tecnologías desarrolladas por el W3C, se ha creado un ecosistema donde los datos son ciudadanos de primera clase, promoviendo una mayor interoperabilidad, precisión y automatización en el manejo de la información. Proyectos como Schema.org y la LOD Cloud demuestran el potencial y la aplicabilidad de estos conceptos en el mundo real, destacando la importancia y el impacto de la Web Semántica en la sociedad actual.

2.3. RDF (Resource Description Framework)

El *Resource Description Framework* (RDF) es un estándar que permite la codificación, intercambio y reusabilidad de metadata estructurada [Miller, 1998], para representar información en la Web, ofreciendo facilidades para lograr el procesamiento automático de recursos web y siendo tan flexible que puede ser aplicado en cualquier área.

RDF fue desarrollado originalmente por el World Wide Web Consortium (W3C) en 1999 como parte de su visión para la Web Semántica [Lassila, 1997]. A lo largo de los años, RDF ha evolucionado significativamente, la versión 1.0 fue lanzada en 2004, introduciendo mejoras en la sintaxis y semántica [Pan, 2009]. En 2014, se lanzó RDF 1.1, que trajo consigo varias actualizaciones importantes, incluyendo la adopción de IRIs (Internationalized Resource Identifiers) en lugar de URIs, y la introducción de nuevos tipos de datos y vocabularios.

2.3.1. Fundamentos de RDF

El modelo o estructura base de RDF es un conjunto de tripletas (ver figura 2) que constan de:

1. **Sujeto o Recurso:** Aquello que se está describiendo. Un sujeto puede ser una página web, un elemento de ella, o incluso un conjunto de páginas. No tiene porque ser un elemento web, también se puede usar para describir elementos físicos como un libro. Un sujeto está representado por un IRI¹ o un nodo en blanco.
2. **Predicado:** Es la propiedad o relación usada para describir un recurso. Al igual que un sujeto, se representa con un IRI.
3. **Objeto:** Es el valor de la propiedad dictada por el predicado, o el otro recurso con el que se establece la relación. Está representado por un IRI, un nodo en blanco, o un literal².

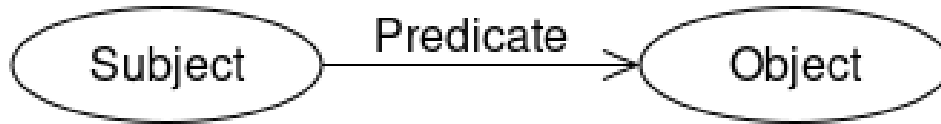


Figura 2: Un grafo de RDF con dos nodos (sujeto y objeto) y un arco que los conecta (predicado).

Fuente: [Richard Cyganiak,].

Las diferencias entre RDF y otros modelos de grafos radican en su formalismo y estandarización para la web. Mientras que otros modelos de grafos pueden ser específicos de una aplicación o dominio, RDF está diseñado para ser interoperable en la web, permitiendo que datos de diferentes fuentes se combinen y se utilicen conjuntamente. En términos matemáticos, un triple RDF puede formalizarse como (s, p, o) , donde s es el sujeto, p el predicado y o es el objeto.

2.3.2. Serialización de RDF

RDF puede ser serializado de varias formas, lo que permite su almacenamiento y transferencia en diferentes formatos según las necesidades específicas.

- **Turtle (Terse RDF Triple Language):** Turtle es una sintaxis amigable para humanos que facilita la escritura y lectura de datos RDF. Ofrece ventajas como la compactación de URIs y la agrupación de triples que comparten el mismo sujeto.

¹Internationalized Resource Identifier, es una generalización de URIs para identificar algo de forma única

²Strings Unicode, para representar cadenas de texto

- **N-Triples:** N-Triples es un formato simple y lineal para representar datos RDF. Es particularmente útil para el procesamiento y almacenamiento de grandes volúmenes de datos debido a su simplicidad.
- **RDF/XML:** Este formato utiliza XML para representar datos RDF, lo que lo hace adecuado para aplicaciones que ya utilizan XML y necesitan integrar datos RDF. Es más verboso que Turtle y N-Triples, pero su uso es conveniente en contextos donde XML es el formato estándar.

2.3.3. Vocabularios RDF y RDF Schema

RDF Schema (RDFS) es una extensión de RDF que proporciona un vocabulario para describir propiedades y clases de recursos RDF, permitiendo la creación de esquemas y ontologías. Los vocabularios comunes en RDF incluyen términos que ayudan a definir las relaciones semánticas entre recursos. Por ejemplo:

- **Dublin Core:** Un conjunto de términos para describir recursos de información.
- **FOAF (Friend of a Friend):** Un vocabulario para describir personas y sus relaciones.
- **SKOS (Simple Knowledge Organization System):** Un modelo para representar sistemas de clasificación y taxonomías.

RDFS permite definir esquemas y ontologías que estructuran datos de manera coherente y significativa. Por ejemplo, se pueden definir clases como *Person* y propiedades como *knows* para representar relaciones entre personas.

2.3.4. Ejemplo de RDF

Un ejemplo de RDF de manera informal (sin utilizar los identificadores únicos) es el siguiente: En la figura 3 se presenta un conjunto de tripletas que describen información acerca de Bob y La Mona Lisa, en la figura 4 se muestra el grafo que representa lo expresado en las tripletas.

RDF proporciona una base sólida para la representación de datos en la web, facilitando la interoperabilidad y la reutilización de la información. Sus diversas formas de serialización y la capacidad de definir vocabularios y esquemas hacen de RDF una herramienta poderosa para la Web Semántica.

```

<Bob> <is a> <person>.
<Bob> <is a friend of> <Alice>.
<Bob> <is born on> <the 4th of July 1990>.
<Bob> <is interested in> <the Mona Lisa>.
<the Mona Lisa> <was created by> <Leonardo da Vinci>.
<the video 'La Joconde à Washington'> <is about> <the Mona Lisa>

```

Figura 3: Tripletas RDF (informales)

Fuente: [Guus Schreiber,].

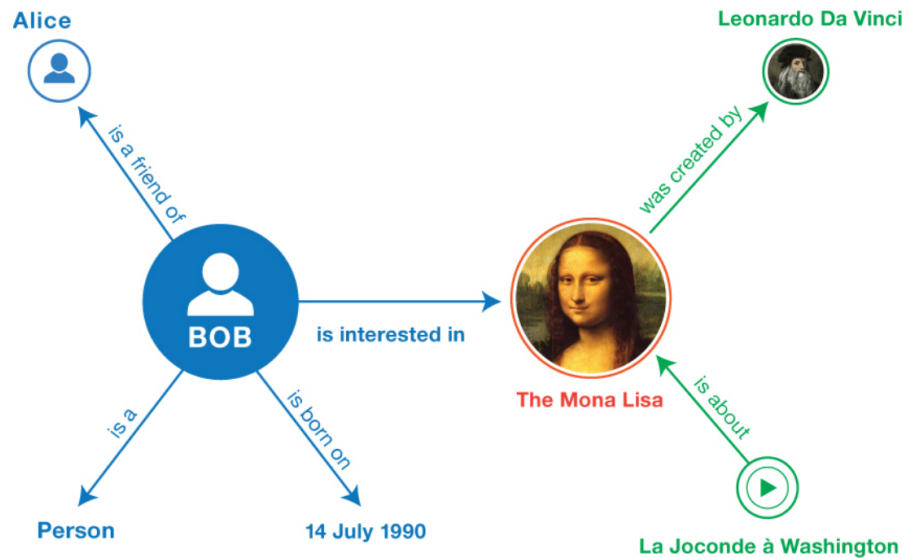


Figura 4: Grafo de las tripletas de la figura 3

Fuente: [Guus Schreiber,].

2.4. SPARQL

SPARQL es el lenguaje estandarizado para realizar consultas sobre grafos RDF [Steve Harris, 2013]. Fue desarrollado por el W3C y se convirtió en una recomendación oficial en enero de 2008. A lo largo de los años, ha sido actualizado para incluir más funcionalidades y optimizaciones, siendo su versión más reciente SPARQL 1.1, lanzada en marzo de 2013, la cual incluye características avanzadas como subconsultas, agregaciones y negación.

Dentro de las operaciones básicas que ofrece SPARQL para interactuar con datos RDF se tienen:

- **SELECT:** Recupera datos en forma de tablas.
- **CONSTRUCT:** Devuelve un grafo RDF construido a partir de los datos obtenidos en la

consulta.

- **ASK:** Devuelve un valor booleano indicando si existen datos que cumplen con el patrón de la consulta.
- **DESCRIBE:** Recupera un grafo RDF que describe los recursos que cumplen con el patrón de la consulta.

Además se proporcionan diversas funciones y operadores que enriquecen sus capacidades de consulta como *Filter* para restringir los resultados mediante condiciones, *Optional* para incluir datos adicionales en caso que estén disponibles, o *Union* que permite combinar resultados de múltiples patrones.

SPARQL es ampliamente utilizado en aplicaciones de datos abiertos y en la web semántica. Un ejemplo notable es DBpedia, un proyecto que extrae información estructurada de Wikipedia y la hace accesible mediante consultas SPARQL. Un ejemplo de consulta SPARQL en DBpedia es el siguiente:

```
SELECT ?person ?birthDate WHERE {  
  ?person dbo:birthPlace dbr:Berlin.  
  ?person dbo:birthDate ?birthDate.  
}
```

Una característica fundamental de SPARQL son los *basic graph patterns* (BGPs). Un BGP es un conjunto de triples que SPARQL intenta encontrar en el grafo RDF, en donde cada sujeto, predicado y objeto puede ser una variable (demarcados por un “?” al principio). En el ejemplo anterior el BGP es:

```
?person dbo:birthPlace dbr:Berlin.  
?person dbo:birthDate ?birthDate.
```

Este patrón busca todos los sujetos que tengan una propiedad *birthPlace* con el valor “Berlín” y una propiedad *birthDate* con algún valor, es decir, obtiene las personas que nacieron en Berlín junto con su fecha de nacimiento.

A pesar de que RDF tenga datos con formato de grafo dirigido, sus funcionalidades navegacionales son limitadas, efecto que se hace más notorio al considerar el vocabulario de RDFS, en donde revisar condiciones como que un objeto sea una subclase de otro o una subpropiedad de otro, requiere navegar la data en RDF [Pérez *et al.*, 2010]. Un ejemplo de esto es la siguiente consulta que no puede ser expresada en SPARQL sin alguna capacidad navegacional adicional.

Consideremos el grafo de RDF mostrado en la figura 5, el cual contiene información acerca de ciudades, servicios de transporte entre ellas y relaciones entre estos servicios de transporte.

En este caso se tiene el servicio “Seafrance” que es una subpropiedad (*sp*) del servicio “ferry”, que a su vez es una subpropiedad del servicio “transport”. Supongamos que se quiere saber si un par de ciudades *A* y *B* están conectadas por una secuencia de servicios de transporte, sin saber de antemano que servicios forman estas conexiones. Para responder a tal consulta se debe probar si existe un camino que conecta *A* y *B* de tal forma que cada arco del camino está conectado con “transport” al seguir la secuencia de relaciones de subpropiedad. En este caso “Paris” y “Calais” cumplen esta condición, pues el camino consiste en un solo arco de etiqueta “TGV”, etiqueta que corresponde a una subpropiedad de “train” que a su vez es una subpropiedad de “transport”; las condiciones de esta consulta también se mantienen para “Paris” y “Dover”, entre otras.

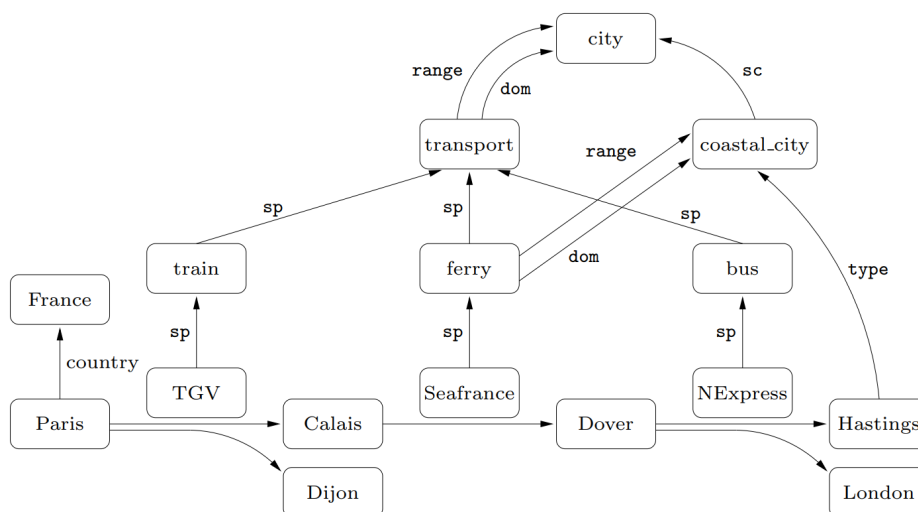


Figura 5: Un grafo RDF que guarda información acerca de servicios de transporte entre ciudades.

Fuente: [Pérez *et al.*, 2010].

2.5. Sistemas de Bases de Datos y Planes de Ejecución

En el ámbito de las bases de datos, existen diferencias clave entre los sistemas de bases de datos relacionales y los de grafos. Las bases de datos relacionales (RDBMS) organizan los datos en tablas con filas y columnas, y utilizan SQL (Structured Query Language) para la manipulación y consulta de datos. Por otro lado, como vimos en la sección 2.1, las bases de datos de grafos están diseñadas para almacenar y consultar datos que están interconectados y mejor representan las relaciones entre datos usando nodos, aristas y propiedades. Entre los ejemplos de bases de datos de grafo podemos encontrar soluciones comerciales de Oracle, Amazon Neptune o Neo4J, o soluciones de código abierto como Blazegraph, Virtuoso o Apache Jena.

2.5.1. Planificación y Ejecución de Consultas

La planificación y ejecución de consultas en bases de datos implica varias fases clave que transforman una consulta en un plan ejecutable y finalmente en un resultado, en [Kossmann, 2000] se muestra una arquitectura general la cual podemos apreciar en la figura 6 y se descompone en:

1. **Analizador:** La consulta es analizada para verificar su sintaxis y semántica, y se convierte en una representación interna que puede ser procesada por el sistema.
2. **Reescritura:** Transforma la consulta para realizar optimizaciones que son beneficiosas independientemente del estado físico del sistema, como la eliminación de predicados redundantes y la simplificación de expresiones.
3. **Optimización:** Realiza optimizaciones que dependen del estado físico del sistema. Decide qué índices utilizar, qué métodos emplear (e.g., hashing, sorting) y en qué orden ejecutar las operaciones de la consulta.
4. **Refinamiento:** Transforma el plan producido por el optimizador en un plan ejecutable, generando un código similar a ensamblador para evaluar expresiones y predicados eficientemente.

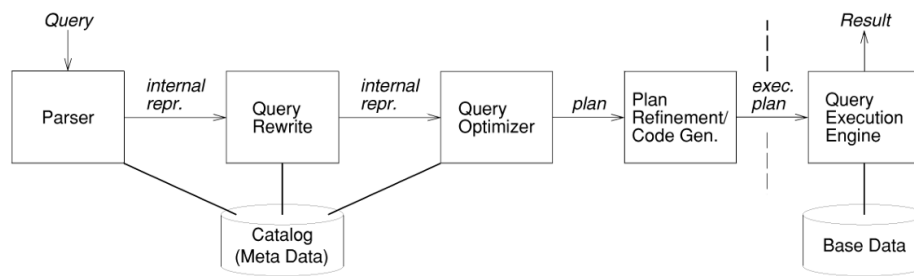


Figura 6: Fases del procesamiento de consultas

Fuente: [Kossmann, 2000].

La optimización de consultas es esencial para reducir la latencia y mejorar el rendimiento del sistema de base de datos. Un plan de ejecución bien optimizado puede minimizar el uso de recursos y el tiempo de respuesta, mejorando la eficiencia general del sistema; por el contrario, un plan de ejecución mal optimizado podría llegar a considerar tiempos de ejecución mucho más altos, haciendo que la ejecución de algunas consultas resulte una tarea imposible.

En los sistemas RDF, la optimización de consultas utiliza estadísticas y cardinalidades para mejorar el rendimiento, proporcionando información sobre la distribución de datos o el número de ocurrencias de ciertos patrones de consulta. Estas métricas ayudan al optimizador de consultas elegir el mejor plan de ejecución.

2.6. Algoritmos de Aprendizaje Automático

El aprendizaje automático (Machine Learning, ML) es una subdisciplina de la inteligencia artificial que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender a partir de datos y hacer predicciones o decisiones basadas en esa información. Los algoritmos de aprendizaje automático se pueden categorizar en dos grandes grupos: aprendizaje supervisado y no supervisado.

2.6.1. Aprendizaje Supervisado

En el aprendizaje supervisado, el algoritmo recibe un conjunto de datos etiquetados como entrada, donde cada entrada está asociada a una salida deseada. El objetivo es que el modelo aprenda a mapear entradas a salidas, de modo que pueda predecir correctamente las salidas para nuevas entradas no vistas. Este tipo de aprendizaje se divide en dos subcategorías principales:

- **Clasificación:** Donde las salidas son categorías discretas. Ejemplos incluyen la detección de spam en correos electrónicos o indicar el tipo de animal presente una imagen.
- **Regresión:** Donde las salidas son valores continuos. Ejemplos incluyen la predicción de precios de viviendas o la estimación de la demanda de productos.

Tom Mitchell definió formalmente el aprendizaje supervisado de la siguiente manera [Mitchell, 1997]: “Un programa de computadora se dice que aprende de la experiencia **E** con respecto a alguna clase de tareas **T** y medida de rendimiento **P**, si su rendimiento en tareas en **T**, medido por **P**, mejora con la experiencia **E**.” Esta definición enfatiza la relación entre la experiencia, las tareas y el rendimiento en el proceso de aprendizaje.

2.6.2. Aprendizaje No Supervisado

En el aprendizaje no supervisado, el algoritmo recibe un conjunto de datos sin etiquetas y debe encontrar estructuras o patrones en los datos. A diferencia del aprendizaje supervisado, no hay salidas conocidas contra las cuales evaluar el rendimiento del modelo. Las técnicas de aprendizaje no supervisado incluyen:

- **Clustering (agrupamiento):** Donde el objetivo es agrupar las entradas en subconjuntos (clusters) basados en la similitud de sus características. Un ejemplo es el algoritmo K-means, que agrupa datos en K clusters.

- **Reducción de dimensionalidad:** Donde el objetivo es reducir el número de variables bajo consideración, obteniendo así un conjunto más manejable de características principales a la vez que no se pierde información relevante. Ejemplos incluyen Análisis de Componentes Principales (PCA) y t-SNE.

2.6.3. Hiperparámetros

Los hiperparámetros son parámetros del modelo que se configuran antes del proceso de entrenamiento y no se ajustan durante este. Incluyen aspectos como la tasa de aprendizaje, el número de capas en una red neuronal, o el número de árboles en un bosque aleatorio (Random Forest). La selección adecuada de hiperparámetros es crucial para el rendimiento del modelo y se puede realizar mediante técnicas como búsqueda en cuadrícula (grid search) o búsqueda aleatoria (random search).

2.6.4. Conjuntos de Entrenamiento, Validación y Prueba

Para entrenar y evaluar modelos de aprendizaje automático, se suelen dividir los datos en tres conjuntos principales:

- **Conjunto de Entrenamiento:** Usado para ajustar el modelo.
- **Conjunto de Validación:** Usado para ajustar los hiperparámetros que afectan al entrenamiento, y usado para prevenir el sobreajuste (overfitting) o subajuste (underfitting).
- **Conjunto de Prueba:** Usado para evaluar el rendimiento final del modelo.

Esta división permite asegurar que el modelo generaliza bien a datos no vistos, ofreciendo una estimación más precisa de su desempeño en el mundo real.

2.6.5. Métricas de Evaluación

Las métricas de evaluación son cruciales para medir el rendimiento de los modelos de aprendizaje automático. Algunas de las métricas comunes incluyen:

- **Error Cuadrático Medio (MSE) y Raíz del Error Cuadrático Medio (RMSE):** Utilizadas para problemas de regresión, estas métricas miden la diferencia promedio entre las predicciones del modelo y los valores reales.
- **Precisión, Recall y F1-Score:** Utilizadas para problemas de clasificación, estas métricas evalúan la exactitud del modelo en la predicción de clases específicas.

El proceso de entrenamiento en el aprendizaje automático implica varias etapas críticas para garantizar que el modelo aprenda de manera efectiva a partir de los datos disponibles. Inicialmente, el modelo se entrena utilizando el conjunto de datos de entrenamiento, donde se ajustan los pesos y parámetros internos del modelo para minimizar una función de error específica. Este error de entrenamiento se utiliza para guiar el proceso de optimización.

Durante el entrenamiento, es esencial evaluar el modelo utilizando el conjunto de validación para ajustar los hiperparámetros y prevenir el sobreajuste (overfitting). El objetivo es lograr que el modelo no solo tenga un buen rendimiento en los datos de entrenamiento, sino que también generalice bien a datos no vistos previamente.

La relación entre la experiencia **E**, las tareas **T** y la medida de rendimiento **P** mencionada por Mitchell se manifiesta en este proceso. La experiencia **E** se refiere a los datos de entrenamiento, las tareas **T** son las predicciones que el modelo debe realizar, y la medida de rendimiento **P** es la métrica utilizada para evaluar la precisión del modelo. La capacidad de generalización del modelo es el desafío central del aprendizaje automático y se mide mediante el rendimiento en el conjunto de prueba, asegurando que el modelo puede manejar entradas nunca antes vistas con eficacia.

2.7. Predicción de latencia

La predicción de latencia es un aspecto crítico en el diseño y operación de sistemas de bases de datos, independientemente de su modelo subyacente. En un mundo donde los datos crecen exponencialmente y las consultas se vuelven cada vez más complejas, la capacidad de estimar con precisión el tiempo que llevará ejecutar una consulta es esencial para garantizar un rendimiento óptimo. Al predecir la latencia, los sistemas pueden tomar decisiones informadas sobre la planificación y optimización de consultas. Esta es una herramienta fundamental para construir sistemas de bases de datos robustos y eficientes.

2.7.1. Predicción de latencia en RDBMS

La predicción de latencia en los Sistemas de Gestión de Bases de Datos Relacionales (RDBMS) ha sido un área de investigación y desarrollo significativa durante décadas. Los RDBMS, como MySQL, PostgreSQL y Oracle, manejan grandes volúmenes de datos y ejecutan consultas complejas, lo que requiere una planificación y optimización cuidadosas para garantizar un rendimiento eficiente.

Un enfoque común para la predicción de latencia en RDBMS implica el uso de estadísticas del sistema, como cardinalidades de tablas, selectividades de predicados y costos de operaciones individuales (e.g., uniones, agregaciones). Estas estadísticas permiten a los optimizadores de consultas generar planes de ejecución óptimos. Técnicas avanzadas, como el

aprendizaje supervisado, se han utilizado para entrenar modelos predictivos que mejoran la estimación de costos y tiempos de ejecución.

2.7.2. Modelos de Predicción en SPARQL

A diferencia de los RDBMS, SPARQL es un lenguaje de consulta diseñado específicamente para grafos RDF. Las bases de datos de grafos presentan características únicas para poder modelar relaciones complejas y datos interconectados, que no están presentes en las bases de datos relacionales tradicionales. Estas diferencias estructurales y funcionales presentan desafíos únicos para la predicción de latencia.

Una de las principales diferencias es que las consultas SPARQL a menudo implican patrones de grafos básicos (BGPs) y navegación a través de grafos, lo que complica la estimación de tiempos de ejecución. Además, las estadísticas disponibles en bases de datos de grafos suelen ser menos detalladas y menos precisas que en RDBMS, lidiar con esta incertidumbre y falta de datos precisos dificulta la creación de modelos predictivos precisos.

Los modelos de predicción de latencia en SPARQL han comenzado a emerger recientemente, aprovechando técnicas de aprendizaje automático para abordar los desafíos específicos de las bases de datos de grafos, los cuales pueden capturar patrones complejos en los datos y proporcionar estimaciones más robustas. Estos modelos buscan predecir el tiempo de ejecución de consultas SPARQL basándose en características extraídas de las consultas y de su estructura.

Es acá donde se desarrolló un modelo basado en convoluciones sobre estructuras de árboles para caracterizar y predecir la latencia de las consultas SPARQL [Casals *et al.*, 2023]. Este modelo considera no solo la estructura de las consultas, sino también la topología del grafo y las características de los datos, lo que permite una predicción más precisa y generalizable. El modelo propuesto está basado en Neo [Marcus *et al.*, 2019], originalmente pensado y utilizado en bases de datos relacionales.

2.7.3. Codificación de consultas

La codificación de la información es un paso crítico en cualquier modelo de aprendizaje automático, una buena codificación es la que entrega la mayor cantidad de información relevante para la tarea en cuestión. Por ejemplo, en el contexto de SPARQL podríamos suponer que el tiempo de latencia se ve afectado por el grafo que se forma a partir del texto de la consulta, por lo que se debería proponer una forma de poder encapsular esta estructura y que pueda ser entendido y procesado por el modelo de aprendizaje.

Una técnica comúnmente utilizada es la codificación One-Hot, donde cada característica se representa como un vector binario. Por ejemplo, la presencia de ciertos operadores SPARQL

(como FILTER, OPTIONAL, UNION) podría codificarse como un vector donde 1 indica presencia y 0 indica ausencia. Esto permite al modelo aprender patrones basados en la ocurrencia de estos elementos.

Para capturar la estructura de árbol de los planes de ejecución de consultas SPARQL, se pueden emplear representaciones específicas. Una opción es utilizar una codificación basada en paréntesis, donde cada nodo del árbol se representa con un par de paréntesis y los nodos hijos se anidan dentro. Por ejemplo, (Join(Scan(A))(Scan(B))). Otra alternativa es usar una representación basada en listas, donde cada nodo se representa como una lista que contiene el tipo de operación y referencias a sus nodos hijos. Estas representaciones permiten preservar la jerarquía y las relaciones entre las operaciones del plan de ejecución.

Además, características numéricas como la selectividad de los predicados, el tamaño de los resultados intermedios, o estadísticas del conjunto de datos pueden requerir escalamiento o normalización para facilitar el aprendizaje del modelo. Técnicas como la normalización min-max, que escala los valores al rango [0, 1], o la estandarización, que centra y escala los datos para tener media cero y desviación estándar uno, pueden aplicarse para asegurar que estas características tengan un rango consistente y evitar que ciertas características dominen sobre otras.

La selección de las codificaciones adecuadas depende de la arquitectura del modelo y del problema específico. Experimentar con diferentes representaciones y evaluar su impacto en el rendimiento del modelo es crucial. El objetivo es encontrar codificaciones que capturen la mayor cantidad de información relevante y permitan al modelo aprender patrones significativos de manera efectiva.

Es importante destacar que la codificación es solo una parte del proceso. También se deben considerar otros aspectos, como la arquitectura del modelo, la selección de hiperparámetros y las técnicas de regularización, para obtener un modelo de predicción de alto rendimiento.

2.8. LSQ 2.0: Un conjunto de datos de consultas SPARQL

El Linked SPARQL Queries (LSQ) 2.0 [Stadler *et al.*, 2024] es un extenso conjunto de datos que ofrece información clave de las consultas SPARQL utilizadas en entornos del mundo real. Desarrollado como una expansión significativa del LSQ original, este dataset abarca más de 43 millones de ejecuciones de consultas, extraídas de los logs de 27 endpoints SPARQL diferentes.

LSQ 2.0 surge como respuesta a una necesidad crítica en la comunidad de investigación de SPARQL: contar con un corpus extenso y diverso de consultas reales para análisis, investigación y evaluación de sistemas. Ya que gran parte de la investigación que se realizaba en torno a RDF se basaba en la creación de conjuntos de datos sintéticos, con consultas creadas manualmente que pueden no reflejar fielmente los patrones de uso real. Dentro de las

características presentes en este dataset tenemos características estáticas de consultas, que son inherentes al texto de la consulta y su equivalencia de representación en forma de grafo: cantidad de *triple patterns* y *basic graph patterns*, *join variables*, etc; además de información sobre la ejecución de las consultas, como el tiempo de ejecución y cantidad de resultados.

LSQ utiliza un modelo consistente con RDF para representar toda esta información (véase figura 7, lo que facilita su análisis y entrega una gran capacidad para procesamiento automatizado. Esta representación permite realizar consultas SPARQL para obtener información sobre las consultas que abarca el dataset, como las que tienen una cantidad de resultados mayor a cero, consultas con una forma en particular, consultas con algún filtro o modificador en específico, etc.

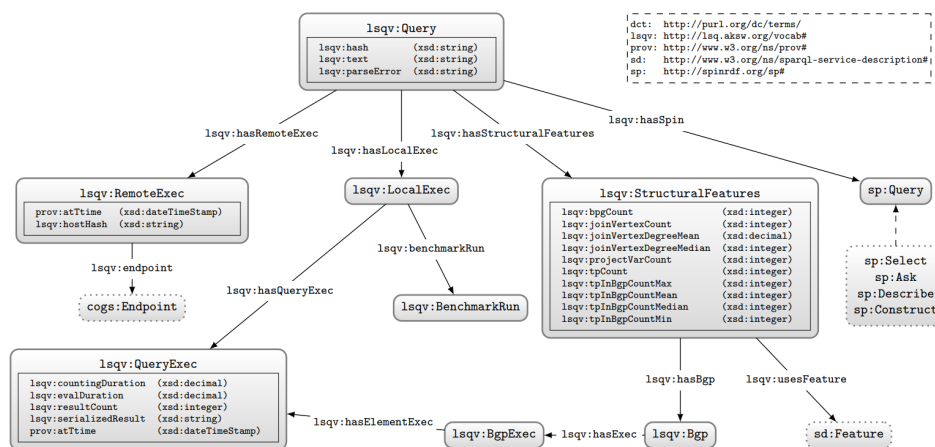


Figura 7: Modelo de datos de LSQ
Fuente: [Stadler et al., 2024].

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

3.1. Modelo de predicción

El modelo de predicción de latencia para consultas SPARQL propuesto en [Casals *et al.*, 2023] se basa en una arquitectura de red neuronal profunda que se inspira en el modelo Neo [Marcus *et al.*, 2019], originalmente diseñado para consultas SQL. La arquitectura del modelo (figura 8) se compone de varias partes clave:

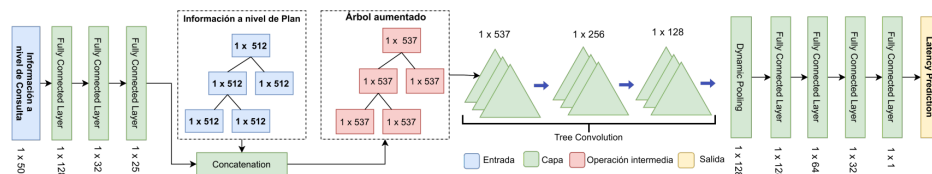


Figura 8: Arquitectura genérica de la red neuronal, las unidades por capas son válidas a modo de ejemplo

Fuente: [Casals *et al.*, 2023].

- **Codificación a nivel de consulta:** Esta parte del modelo procesa características inherentes de la consulta SPARQL, como la presencia de filtros específicos y el valor numérico usado por el operador LIMIT. Estas características se procesan a través de capas densamente conectadas (fully-connected layers).
- **Codificación a nivel de plan de ejecución:** Esta es la parte más innovadora del modelo, que trata el plan de ejecución de la consulta como un árbol. Cada nodo del árbol representa una operación en el plan (como un join), y se codifica con un vector que incluye información sobre el tipo de operación y los predicados involucrados.
- **Convoluciones sobre árboles:** Inspirado en el trabajo de [Mou *et al.*, 2016] sobre redes neuronales convolucionales para estructuras de árbol (TBCNN), esta parte del modelo aplica operaciones de convolución sobre la estructura de árbol del plan de ejecución. Esto permite al modelo aprender patrones en diferentes niveles de granularidad del plan.
- **Agrupación dinámica:** Después de las capas convolucionales, se aplica una operación de agrupación dinámica que colapsa la estructura de árbol en un vector de tamaño fijo, preservando la información más relevante [Mou *et al.*, 2016].
- **Capas finales de predicción:** El vector resultante de la agrupación dinámica se procesa a través de capas densamente conectadas adicionales, culminando en una única neurona de salida que predice el tiempo de ejecución de la consulta.

Este modelo junto con su representación de consultas permite manejar planes de ejecución de tamaño variable, lo cual es esencial dado que las consultas SPARQL pueden variar enormemente en complejidad y tamaño. El modelo se entrena utilizando una función de pérdida basada en el error cuadrático medio (MSE), que penaliza las diferencias entre el tiempo de ejecución predicho y el real.

3.2. Extracción de consultas y características

Como mencionamos en la sección 2.8, el dataset LSQ 2.0 consta de consultas de 27 endpoints distintos, de los cuáles se seleccionaron 3 para su estudio:

1. **DBPedia:** Ofrece una de las colecciones más diversas en términos de clientes y tipos de consultas. Es especialmente rico en consultas que utilizan patrones de propiedad (property paths), lo que lo hace ideal para analizar consultas complejas y estructuradas.
2. **LinkedGeoData:** Este dataset se destaca por contener algunas de las consultas más costosas computacionalmente. Según los análisis, las consultas de LinkedGeoData tienden a correlacionarse con mayores tamaños de resultados y un grado medio de vértice de unión (mean join-vertex degree) más alto. Esto lo convierte en un excelente candidato para estudiar la predicción de latencia en consultas particularmente desafiantes.
3. **KEGG (Kyoto Encyclopedia of Genes and Genomes):** Aunque menos voluminoso que DBpedia o LinkedGeoData, el dataset KEGG presenta tiempos de ejecución relativamente altos. Esto sugiere que contiene consultas complejas o que operan sobre estructuras de datos intrincadas, ofreciendo así otro ángulo valioso para el estudio de la predicción de latencia.

La inclusión de estos tres datasets permitirá analizar un espectro amplio de complejidades de consulta, además de evaluar cómo diferentes dominios de información afectan el modelo de predicción, probando la robustez y generalización del modelos de predicción en contextos variados.

Dado que la información está modelada en RDF, podemos realizar una consulta para obtener todas las consultas que cumplan alguna condición, en este caso nos centraremos en aquellas de tipo “SELECT” que tienen un tiempo de ejecución mayor o igual a 2 segundos, a modo de filtrar aquellas consultas que son muy rápidas y no son de real interés en la predicción a comparación de aquellas que toman un tiempo considerable.

PREFIX owl: <<http://www.w3.org/2002/07/owl#>>

PREFIX rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>

```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX lsq: <http://lsq.aksw.org/vocab#>
PREFIX sp: <http://spinrdf.org/sp#>

SELECT DISTINCT *
{
?s lsq:text ?query ;
  lsq:hasSpin ?spin ;
  lsq:hasStructuralFeatures ?features ;
  lsq:hasLocalExec ?exec .
?spin a sp:Select .
?features lsq:joinVertexDegreeMean ?joinVertexDegreeMean ;
  lsq:joinVertexDegreeMedian ?joinVertexDegreeMedian ;
  lsq:bgpCount ?bgpCount ;
  lsq:tpCount ?tpCount ;
  lsq:tpInBgpCountMean ?tpInBgpCountMean ;
  lsq:tpInBgpCountMin ?tpInBgpCountMin ;
  lsq:tpInBgpCountMedian ?tpInBgpCountMedian ;
  lsq:tpInBgpCountMax ?tpInBgpCountMax ;
  lsq:joinVertexCount ?joinVertexCount ;
  lsq:projectVarCount ?projectVarCount .
?exec lsq:hasQueryExec ?execDetails .
?execDetails lsq:resultCount ?resultSize ;
  lsq:evalDuration ?time .
  FILTER (?time >= 2)
}

```

Con la ejecución de esta consulta obtenemos el texto original de la consulta, el tiempo de ejecución, cantidad de resultados y otras características adicionales que no fueron consideradas en el modelo de predicción original. Los resultados de la ejecución de esta consulta en los diversos datasets se presentan en la tabla 1, en donde podemos ver que más de un 50 % de las consultas obtenidas de DBPedia no poseen resultados.

Dataset	Cantidad de consultas	Cantidad de consultas con resultados >0
DBPedia	9427	5349
LinkedGeoData	3320	3050
KEGG	911	909
Total	13658	9308

Tabla 1: Cantidad de consultas obtenidas

El modelo de predicción considera una codificación a nivel de plan de ejecución, esto no es algo representable en el dataset LSQ pues depende del optimizador en particular, por lo que es necesario generarlo a partir de la consulta original. Para esto se utilizó Apache Jena, el cual

permite compilar una consulta cualquiera y obtener un plan de ejecución, cabe destacar que las consultas originalmente fueron ejecutadas en Virtuoso, por lo que el plan generado por Jena puede no corresponder con el utilizado en la ejecución de la consulta, por lo que se considera esto como una aproximación que puede ser mejorable si se tuviera acceso al plan de ejecución real.

Lo último faltante es la selectividad de los predicados, si bien el modelo de LSQ entrega información sobre la selectividad de los diversos *triple patterns* ejecutados, no existe información explícita sobre el orden en que fueron ejecutadas estas sub-consultas o su posición en el plan de ejecución. Por esto, se decidió simplemente optar por contar los predicados de cada dataset mediante la siguiente consulta:

```
SELECT ?predicate (COUNT(*) AS ?count)
WHERE {
  ?s ?predicate ?o
}
GROUP BY ?predicate
ORDER BY DESC(?count)
```

Dentro de las características extraídas sobre las consultas que nos proporciona el modelo de LSQ tenemos:

- **bgpCount:** Se refiere a la cantidad de Basic Graph Patterns de una consulta.
- **joinVertex:** Número de vértices o nodos involucrados en uniones o joins en la consulta, producido por variables compartidas en distintos triple patterns.
- **joinVertexDegree(Mean,Median):** Grado de vértices involucrados en joins, el grado de un vértice se refiere a cuantas conexiones tiene con otras partes de la consulta. Se tienen 2 características que definen el promedio y mediana.
- **projectVarCount:** Cantidad de variables proyectadas en la consulta, es decir, aquellas que aparecen en una consulta SELECT y son retornadas.
- **tpCount:** Número de triple patterns de una consulta.
- **tpInBgpCount(Max,Mean,Median,Min):** Número de triple patterns contenidos en un BGP dentro de la consulta. Se tienen 4 características que definen el promedio, mediana, mínimo y máximo. El promedio sirve para medir la complejidad típica de los BGPs de la consulta, mientras que el mínimo y máximo nos puede indicar el BGP más simple y más complejo presente en una consulta.

3.3. Exploración de datos

Los datasets generados en la sección anterior se separan en conjunto de entrenamiento/validación y conjunto de pruebas, con una proporción de 70 % y 30 % respectivamente. El conjunto de pruebas se deja completamente aislado, cualquier análisis o modificaciones realizadas serán sobre el conjunto de entrenamiento.

3.3.1. Distribución de tiempo

En las figuras 9, 10, 11 y 12 se muestran histogramas de la distribución de los tiempos de ejecución de consultas en los diferentes conjuntos de datos, utilizando una escala logarítmica en el eje y. Se puede apreciar que la distribución es asimétrica, con una alta concentración de consultas con tiempos de ejecución cortos, menores a 10 segundos, y algunos picos en diferentes intervalos de tiempo. La cantidad de consultas disminuye considerablemente para tiempos más altos, en el rango de 50-60 segundos.

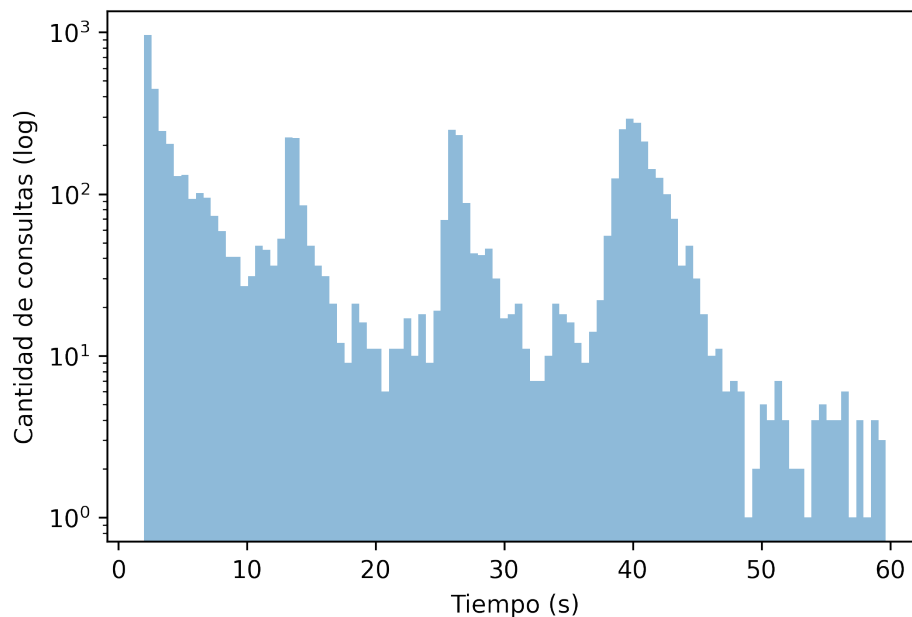


Figura 9: Distribución de datos de conjunto de entrenamiento de dataset DBPedia
Fuente: Elaboración Propia.

3.3.2. Correlaciones

En la figura 13 se muestra un mapa de calor de correlaciones entre las diferentes características de las consultas, dado el caso de uso, nos interesa en particular la correlación que tiene

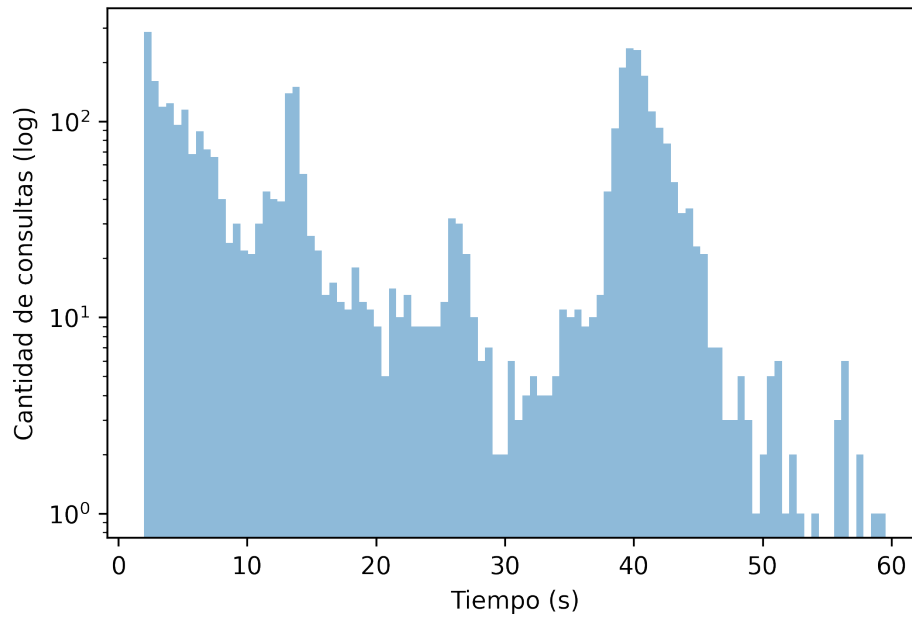


Figura 10: Distribución de datos de conjunto de entrenamiento de dataset DBPedia, filtradas las consultas que no devuelven resultados. Fuente: Elaboración Propia.

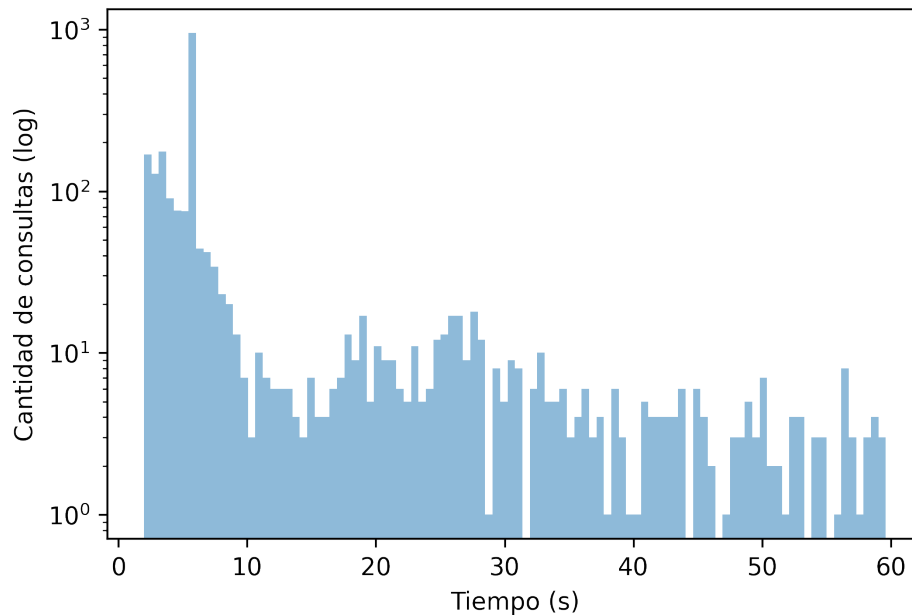


Figura 11: Distribución de datos de conjunto de entrenamiento de dataset LinkedGeoData. Fuente: Elaboración Propia.

con el tiempo de ejecución, se obtiene un valor negativo para todas las variables, lo que indica que las variables son inversamente proporcionales, mientras mayor sea la magnitud de este número, mayor será la relación que tienen ambas variables. En este caso projectVar-

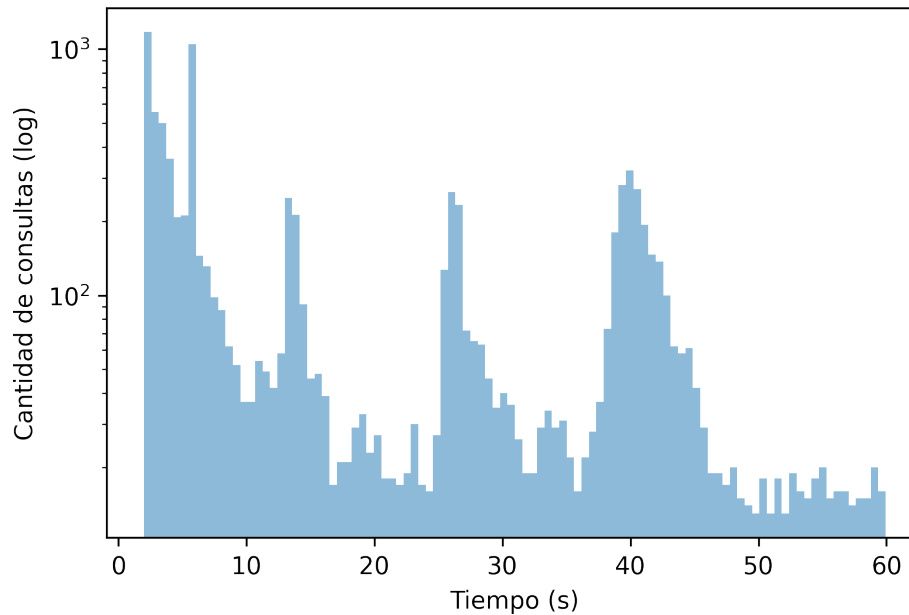


Figura 12: Distribución de datos de conjunto de entrenamiento de dataset combinado: DB-Pedia + LinkedGeoData + KEGG. Fuente: Elaboración Propia

Count es la característica con mayor correlación con el tiempo de ejecución

En la figura 14 se muestra el gráfico de dispersión o *scatter plot* del tiempo de ejecución vs *projectVarCount*, si bien el índice de correlación nos sugiere alguna relación moderada entre ambas variables, no es evidenciada en el gráfico, el caso se repite para cualquier variable. La falta de un patrón obvio en este gráfico nos confirma que la relación entre las características con el tiempo de ejecución es compleja y no puede capturada por una simple correlación o regresión lineal.

Por otra parte, tenemos las correlaciones entre las demás variables o características de las consultas que nos indican una alta correlación entre varias de ellas, estos pares altamente relacionados probablemente indique una redundancia en la información que entregan para el modelo, a la vez que inducen un mayor nivel de complejidad de manera innecesaria al mismo.

3.3.3. Información Mutua

Otro análisis que se puede hacer para medir la relación entre variables es el cálculo de la información mutua. A diferencia de la matriz de correlación, que mide relaciones lineales, la información mutua puede capturar relaciones no lineales entre variables, lo que la hace especialmente útil en casos donde las interacciones son más complejas.

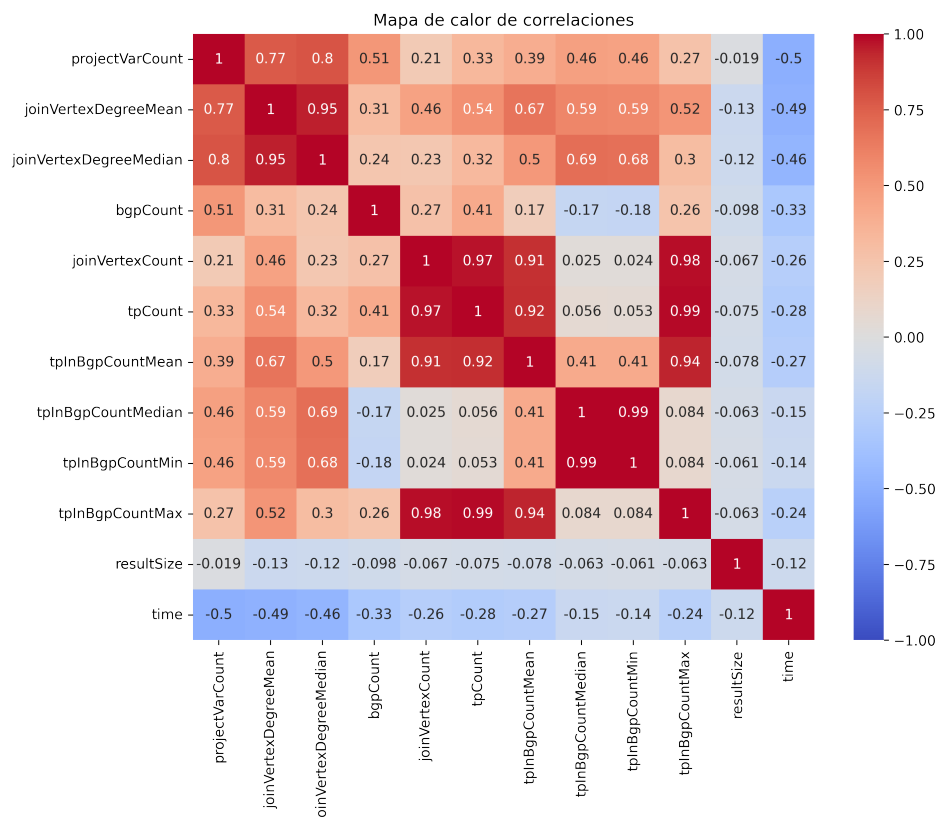


Figura 13: Correlación entre variables del dataset DBPedia

Fuente: Elaboración Propia.

La información mutua cuantifica la cantidad de información que una variable proporciona sobre otra casi de la misma manera que el índice de correlación presentado en la sección 3.3.2, con valores que oscilan entre 0 y 1, donde 0 indica independencia total entre las variables y 1 sugiere una relación perfecta.

En la figura 15 se presentan los valores de información mutua entre las distintas características, observamos que variables como “tpCount” y “projectVarCount” muestran valores de información mutua alrededor de 0.7 con respecto a la variable objetivo de tiempo de ejecución, indicando una fuerte relación no necesariamente lineal. Por otro lado, variables como “tpInBgpCountMin” y “resultSize” presentan valores más bajos, alrededor de 0.1-0.2, sugiriendo una relación más débil.

Es importante contrastar estos resultados con los obtenidos en la matriz de correlación. Mientras que la correlación puede haber mostrado relaciones lineales fuertes o débiles, la información mutua nos permite identificar relaciones más complejas. Por ejemplo, una variable que muestre baja correlación lineal podría tener un valor de información mutua alto, indicando una relación no lineal significativa.

La combinación de ambos análisis nos proporciona una visión más completa de las relaciones

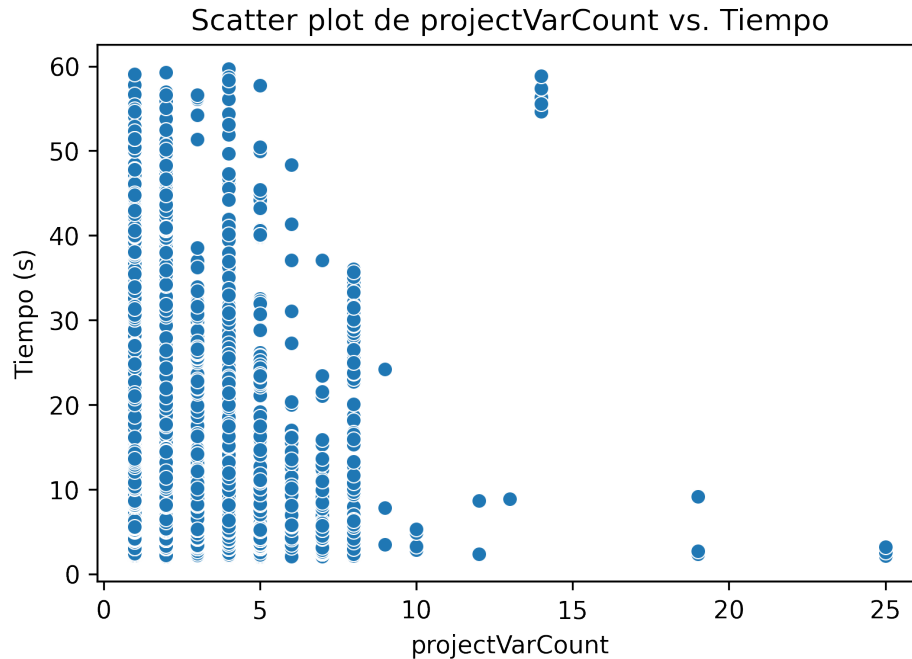


Figura 14: Diagrama de dispersión entre projectVarCount y tiempo de ejecución del dataset DBPedia

Fuente: Elaboración Propia.

entre nuestras variables. Mientras que la correlación nos ayuda a entender las tendencias lineales, la información mutua nos permite capturar patrones más sutiles o complejos que podrían ser de gran aporte para el modelo predictivo. Considerando ambos análisis de correlación, se recomienda:

1. Variables a mantener:

- projectVarCount: Alta información mutua con time y relativamente bajo índice de correlación con otras variables.
- tpCount: La más alta información mutua con "time", aunque redundante con otras variables.
- bgpCount: Información mutua moderada con "time" y baja redundancia con otras variables.
- joinVertexDegreeMean: Con un valor moderado de información mutua, se decide mantener esta variable pues puede entregar información útil sobre la presencia de variables involucradas en Joins.

2. Variables a descartar:

- joinVertexCount: Altamente redundante con tpCount.

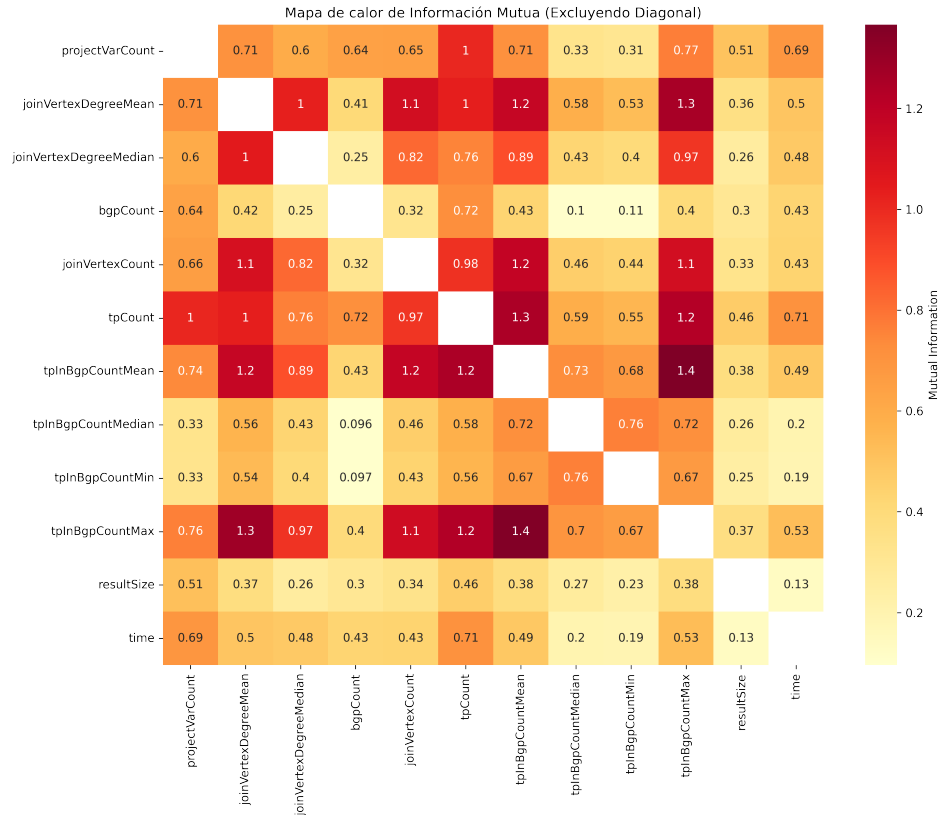


Figura 15: Información mutua entre variables del dataset DBpedia
Fuente: Elaboración Propia.

- tpInBgpCount(Mean/Median/Min/Max): Altamente redundante entre ellas y baja información mutua con respecto a objetivo, se prefiere tpCount en cualquier caso, aunque se podría hacer una excepción y mantener tpInBgpCountMax que es la que tiene una mayor información mutua.
- resultSize: Confirmada su baja relevancia tanto en correlación como en información mutua.
- joinVertexDegreeMedian: Se prefiere joinVertexDegreeMean.

3.4. Experimentos

Para evaluar el modelo con la información obtenida, se consideran dos experimentos base, usando la totalidad de las consultas de los datasets y filtrando aquellas que no se obtienen resultados, en los conjuntos de datos de DBpedia, LinkedGeoData y un dataset combinado que considera DBpedia, LinkedGeoData y KEGG. Dado el poco volumen de datos obtenido del endpoint KEGG, se omite su evaluación de forma independiente. Hay que tener en consideración que al hacer una combinación de datasets la selectividad de los predicados se

verá drásticamente afectada, ya que esta corresponde a la cantidad de tripletas que utilizan un predicado en particular dentro de todo el conjunto de datos, y estos 3 conjuntos de datos son completamente independientes, resultando en información no fidedigna, esto nos podría indicar el efecto que tiene esta característica en el modelo de predicción.

Por otra parte, se considera que el volumen de datos puede no ser lo suficientemente grande para entrenar correctamente el modelo, esto sumado al desbalanceo que existe en la distribución de tiempos de consulta visto en la sección 3.3 motiva probar técnicas de balanceo, especialmente considerando que las consultas que más valor aporta predecir correctamente son aquellas con tiempos de ejecución mayores.

Para los experimentos se consideró **over-sampling**, que consiste en aumentar la cantidad de muestras que son consideradas más raras. Para esto se utilizó repetición de aquellas consultas menos representadas, en más detalle: se divide el dataset en una cantidad de bins determinados, 10 y 20, se determina un límite porcentual para los bins con baja representación, se utilizó valores desde 30 a 60 con aumentos de 10, toda consulta que se considere parte de estos bins poco representados será copiada una cantidad de veces que depende cuán representado está el bin respectivo, haciendo que cada copia varíe levemente su tiempo de ejecución, considerando un valor al azar entre 95 % y 105 % del tiempo original.

En la tabla 2 se muestra la cantidad de consultas obtenidas luego de realizar el proceso descrito en el dataset combinado, en la figura 16 se muestra la nueva distribución de los datos para el caso de 10 bins y 60 % de límite. Se puede apreciar un aumento considerable de aquellas consultas menos representadas las cuales se encontraban en los rangos descritos previamente, pasando de un tiempo de ejecución promedio de 19.13 segundos a 22.29, moviendo la mediana de 13.14 a 17.62 segundos.

	30 %	40 %	50 %	60 %
10 Bins	9634	9877	10455	11340
20 Bins	9633	9841	10084	10873

Tabla 2: Cantidad de consultas luego de realizar over-sampling a dataset DBPedia + Linked-GeoData + KEGG

Se realizarán 100 ejecuciones en los distintos datasets a evaluar para cada caso descrito previamente, para tener una comparativa con otro modelo de predicción se utilizará la máquina de soporte vectorial con kernel NuSVR propuesta en [Hasan y Gandon, 2014]. Las métricas a utilizar para evaluar los modelos son:

1. **RMSE (Root Mean Squared Error):** Es la raíz cuadrada del error cuadrático medio, que mide la magnitud promedio del error entre las predicciones del modelo y los valores reales. Penaliza más fuertemente los errores grandes, por lo que es útil cuando los errores grandes son particularmente indeseables.
2. **MAE (Mean Absolute Error):** Es el error absoluto medio, que representa la media de las diferencias absolutas entre las predicciones y los valores reales. Es una métrica

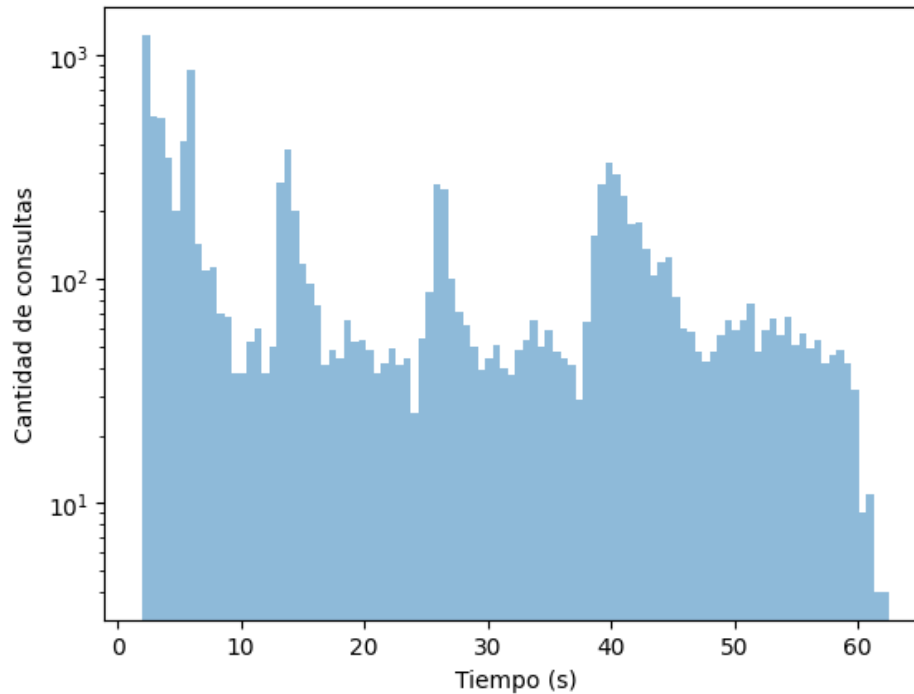


Figura 16: Distribución de datos de conjunto de entrenamiento de dataset combinado: DB-Pedia + LinkedGeoData + KEGG luego de realizar over-sampling con 10 bins y un límite de 60 %

Fuente: Elaboración Propia.

más robusta ante outliers y fácil de interpretar, ya que indica el error promedio en las mismas unidades de los datos.

3. **R2 (Coeficiente de Determinación):** Mide qué proporción de la varianza total de los datos es explicada por el modelo. Varía entre 0 y 1, donde un valor de 1 indica un modelo perfecto y 0 sugiere que el modelo no explica nada la varianza.
4. **MAPE (Mean Absolute Percentage Error):** Es el error porcentual absoluto medio, que calcula el error promedio como un porcentaje de los valores reales. Es útil para interpretar la precisión del modelo en términos relativos, aunque puede ser sensible a valores pequeños en los datos.

Estas métricas han sido seleccionadas porque proporcionan una visión integral de la precisión del modelo desde diferentes perspectivas. RMSE y MAE permiten cuantificar el error promedio, con RMSE enfocándose en los errores grandes y MAE siendo más robusta frente a outliers. El coeficiente de determinación (R2) ayudará a entender qué tan bien el modelo explica la variabilidad de los datos, mientras que MAPE ofrecerá una interpretación en términos porcentuales, lo que facilita la comparación en diferentes escalas. Conjuntamente, estas métricas proporcionarán un análisis exhaustivo del rendimiento y la precisión del modelo en el contexto de los datos disponibles.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

En esta sección se presentan y analizan los resultados obtenidos a partir de los experimentos descritos en la sección 3.4, evaluando el desempeño del modelo utilizando las métricas definidas. Además, se realizará una comparación con el modelo de NuSVR propuesto en [Hasan y Gandon, 2014].

4.1. Resultados y evaluación

Se utilizarán gráficos de dispersión para mostrar la comparativa entre las predicciones realizadas y su valor real, al igual que en [Casals *et al.*, 2023] se seguirá un esquema de colores para categorizar las predicciones:

- Los puntos en color verde representan consultas cuya predicción se considera buena:
 $|y - y'| \leq y * 0,2$
- Los puntos en color azul representan consultas cuya predicción se considera aceptable:
 $|y - y'| \leq y * 0,4$
- El resto de puntos rojos corresponde con predicciones malas.

4.1.1. DBPedia

Los resultados muestran que el modelo de predicción propuesto logra un desempeño notablemente superior al modelo NuSVR en todas las métricas de evaluación consideradas (ver Anexo). En términos de RMSE el modelo propuesto obtiene valores en torno a 9.3 en el dataset regular de DBPedia, mientras que NuSVR tiene un RMSE de alrededor de 20.3, más del doble. Esto indica que el modelo propuesto realiza predicciones mucho más cercanas a los valores reales de tiempo de ejecución.

Analizando el coeficiente de determinación R², el modelo propuesto alcanza valores cercanos a 0.66 en el dataset regular, lo que significa que explica alrededor del 66 % de la varianza en los tiempos de ejecución. En contraste, NuSVR tiene un R² negativo de -0.618, lo que sugiere que su capacidad predictiva es extremadamente pobre, peor incluso que simplemente predecir la media. Respecto a MAPE, si bien ambos modelos tienen valores altos, el modelo propuesto se desempeña mejor, con un MAPE de 42.9 % versus 215 % para NuSVR en el dataset regular.

Los gráficos de dispersión de valores reales vs predichos también evidencian la superioridad del modelo propuesto. Una proporción mucho mayor de las predicciones caen en el rango "bueno/aceptable". Para el dataset regular, el 57.7 % de las predicciones son buenas y 16.9 % aceptables, mientras que para NuSVR sólo el 11.4 % son buenas.

En cuanto al impacto de las técnicas de over-sampling, los resultados son mixtos y las mejoras no parecen ser sustanciales. En algunos casos hay leves reducciones en el error (por ejemplo RMSE bajando de 9.31 a 9.23 con over-sampling de 20 bins al 60 %), pero en otros el error aumenta. Dado que el over-sampling se realizó sólo en el conjunto de entrenamiento, una posible causa es que introdujo cierto sobreajuste, al enfatizar demasiado los casos menos frecuentes.

El modelo propuesto demuestra una clara ventaja predictiva sobre NuSVR en el dataset de DBPedia, logrando errores sustancialmente menores y explicando una porción significativa de la varianza en los tiempos de ejecución de las consultas SPARQL. Las técnicas de over-sampling tuvieron un impacto limitado y posiblemente negativo, lo que sugiere que el desbalance de clases no es el factor determinante en el desempeño del modelo.

er and history, RMSE Train: 8.911816471050175, RMSE VAL: 13.113384299172436, Epoc

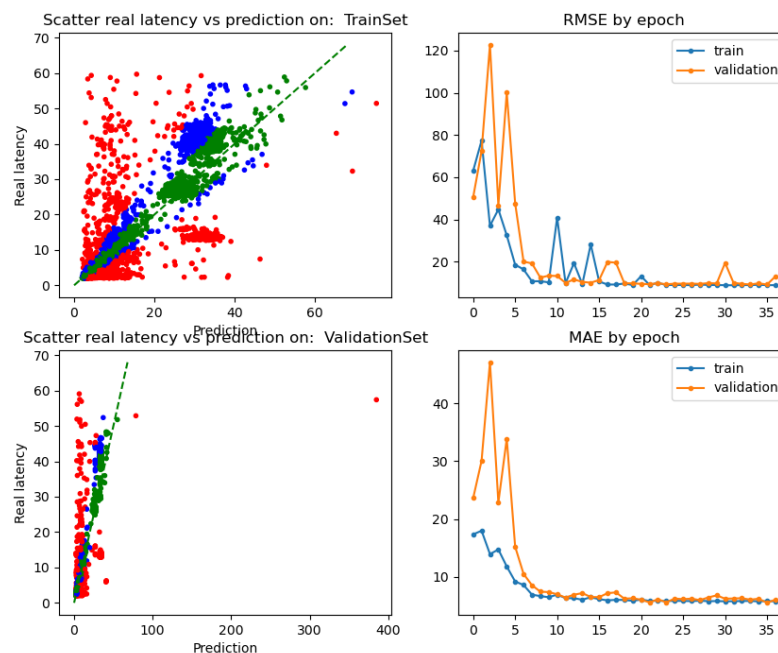


Figura 17: Entrenamiento en DBPedia
Fuente: Elaboración Propia.

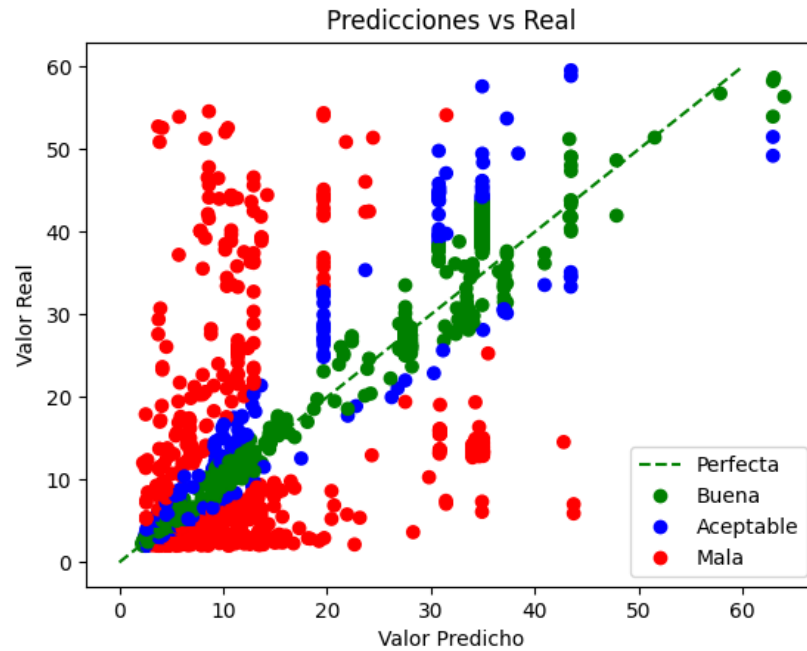


Figura 18: Predicciones en DBPedia

Fuente: Elaboración Propia.

	Buenas	Aceptables	Malas
Total	60.28 %	11.24 %	28.48 %
≥ 20 segundos	84.54 %	7.10 %	9.36 %
< 20 segundos	40.66 %	14.74 %	44.60 %

Tabla 3: Porcentaje de predicciones según color/categoría en DBPedia

	Buenas	Aceptables	Malas
Total	11.38 %	18.70 %	69.92 %
≥ 20 segundos	14.12 %	28.78 %	57.10 %
< 20 segundos	9.08 %	10.20 %	80.72 %

Tabla 4: Porcentaje de predicciones según color/categoría en DBPedia con NuSVR

4.1.2. LinkedGeoData

En cuanto al dataset LinkedGeoData, los resultados muestran algunas similitudes y diferencias interesantes en comparación con DBPedia. Acá las técnicas de over-sampling parecen tener un impacto positivo un poco más notorio. El dataset regular tiene un RSME medio de 10.279 y la mejor configuración (10 bins, límite 40 %) lo reduce a 9.45, con otras configuraciones con menor efecto. Esta diferencia la podemos notar en las tablas 5 y 6, que implica una mejora considerable donde las predicciones malas se reducen en alrededor de un 10 %.

er and history, RMSE Train: 10.435316532086999, RMSE VAL: 11.44168645899246, Epoc

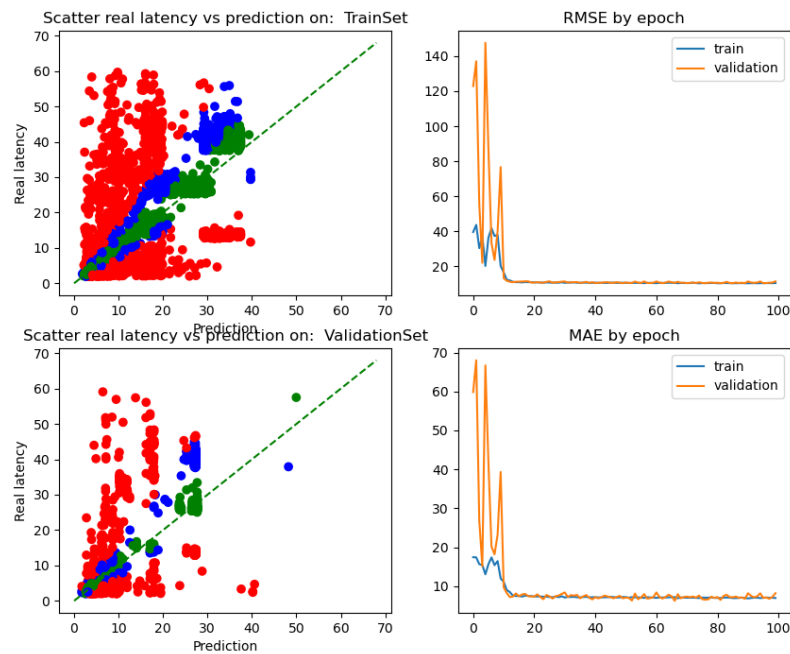


Figura 19: Entrenamiento en DBPedia con NuSVR

Fuente: Elaboración Propia.

Una diferencia notable es que el modelo propuesto alcanza un R^2 de 0.273 en el dataset regular de LinkedGeoData, bastante menor al 0.661 logrado en DBPedia. Esto sugiere que hay una porción mayor de la varianza en los tiempos de ejecución que no está siendo capturada por las características utilizadas.

Otro aspecto distintivo es que, según la tabla 5 de porcentajes por categoría, el modelo se desempeña mejor para consultas de menos de 20 segundos en LinkedGeoData, mientras que en DBPedia era mejor para consultas de más de 20 segundos. Esto podría indicar diferencias en la distribución y características de las consultas lentas en cada dataset.

En resumen, el modelo propuesto mantiene buenos resultados pero hay más varianza sin explicar. Las técnicas de over-sampling tienen un impacto positivo, similar a lo observado en DBPedia. Sin embargo, el modelo muestra un comportamiento diferente en términos de su precisión relativa para consultas rápidas y lentas, lo que sugiere que las características de las consultas difieren entre los datasets.

Estas observaciones refuerzan la noción de que el modelo propuesto es una base sólida para la predicción de latencia de consultas SPARQL, pero también destacan la importancia de considerar las peculiaridades de cada dataset y ajustar el modelo en consecuencia.

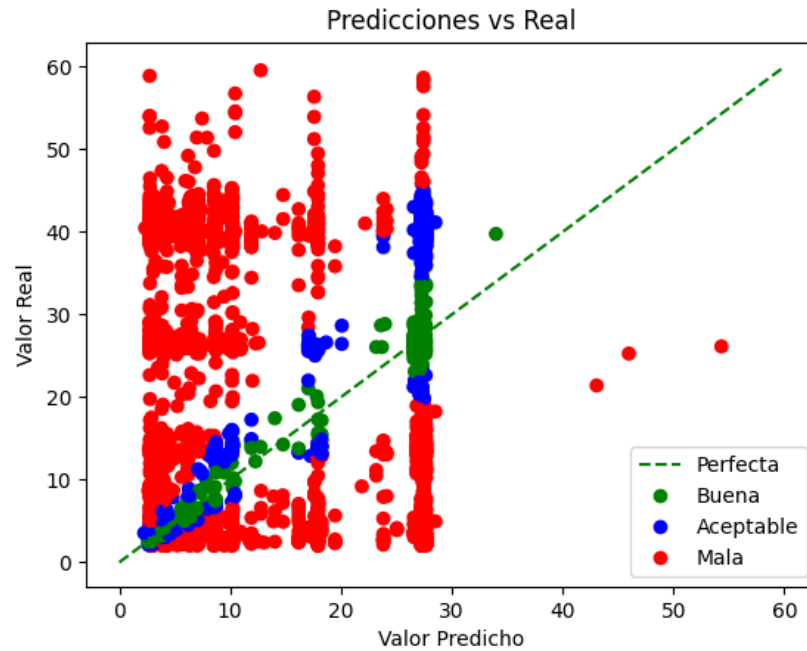


Figura 20: Predicciones en DBPedia con NuSVR
Fuente: Elaboración Propia.

	Buenas	Aceptables	Malas
Total	59.93 %	13.86 %	26.21 %
≥ 20 segundos	26.57 %	21.68 %	51.75 %
< 20 segundos	66.53 %	12.31 %	21.16 %

Tabla 5: Porcentaje de predicciones según color/categoría en LinkedGeoData para dataset regular

Fuente: Elaboración Propia

	Buenas	Aceptables	Malas
Total	66.97 %	13.51 %	19.52 %
≥ 20 segundos	31.47 %	33.57 %	34.96 %
< 20 segundos	74.00 %	9.54 %	16.46 %

Tabla 6: Porcentaje de predicciones según color/categoría en LinkedGeoData para dataset aumentado con 10 bins al 40 %

Fuente: Elaboración Propia

4.1.3. DBPedia + LinkedGeoData + KEGG

Para el dataset combinado (DBPedia + LinkedGeoData + KEGG), los resultados muestran un desempeño sólido del modelo propuesto, a pesar de los desafíos inherentes a la combina-

ter and history, RMSE Train: 8.193754522983813, RMSE VAL: 8.114416785526704, Epoch

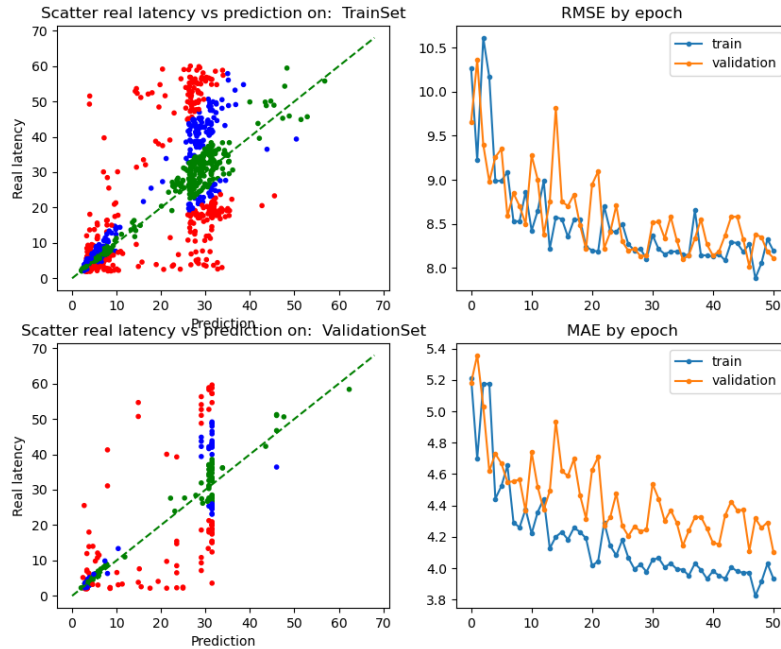


Figura 21: Entrenamiento en LinkedGeoData

Fuente: Elaboración Propia.

ción de datasets diversos. Los valores obtenidos en las diversas métricas son comparables a los obtenidos en DBPedia y ligeramente mejores que en LinkedGeoData. Esto sugiere que el modelo es capaz de manejar efectivamente la heterogeneidad introducida por la combinación de datasets.

Se decidió omitir los experimentos de over-sampling en el dataset combinado dado el impacto limitado y mixto observado en los datasets individuales.

Métrica	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
RMSE	9.742	10.005	1.797	9.151	22.231
MAE	5.929	5.976	0.410	5.369	7.755
R2	0.654	0.623	0.207	-0.803	0.695
MAPE	40.460	40.628	3.737	35.782	61.817

Tabla 7: Métricas de error de predicciones en conjunto de test para dataset combinado

Fuente: Elaboración propia

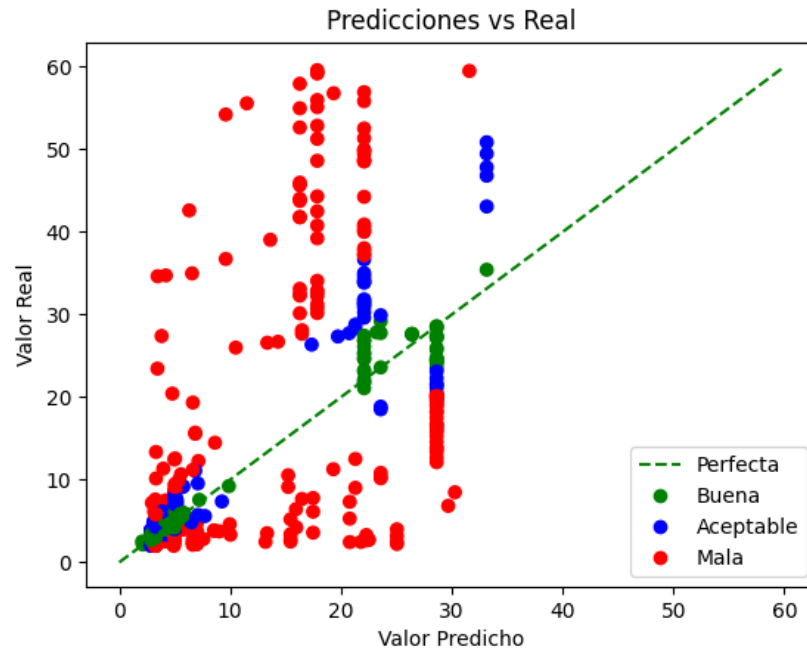


Figura 22: Predicciones en LinkedGeoData para dataset regular
Fuente: Elaboración Propia.

	Buenas	Aceptables	Malas
Total	58.20 %	14.21 %	27.60 %
≥ 20 segundos	71.10 %	15.58 %	13.31 %
< 20 segundos	49.07 %	13.23 %	37.70 %

Tabla 8: Porcentaje de predicciones mejor ejecución según color/categoría en dataset combinado: DBPedia + LinkedGeoData + KEGG

4.2. Generalización entre datasets

En los resultados presentados se puede apreciar una diferencia no menor al evaluar el modelo de predicción en el dataset compuesto por los 3 endpoints, por lo que surge la interrogante sobre cómo se comportaría un modelo al ser entrenado bajo un dataset en particular y evaluado en uno distinto. Para esto se utilizó el mejor modelo entrenado con LinkedGeoData y se evaluó en el conjunto de pruebas de DBPedia.

En este caso los resultados fueron decepcionantes, con un RMSE muy variable que va desde 25 y fracción en el mejor de los casos y una mediana de 714, lo cuál es excesivamente peor comparando con el error obtenido al entrenar y predecir dentro del mismo dataset. Esto nos puede dar como indicio que la selectividad de los predicados es un factor muy influyente para generar una buena predicción, la cual depende completamente de las tripletas que componen el conjunto de datos.

ter and history, RMSE Train: 8.534530006139114, RMSE VAL: 9.265247757089961, Epoch

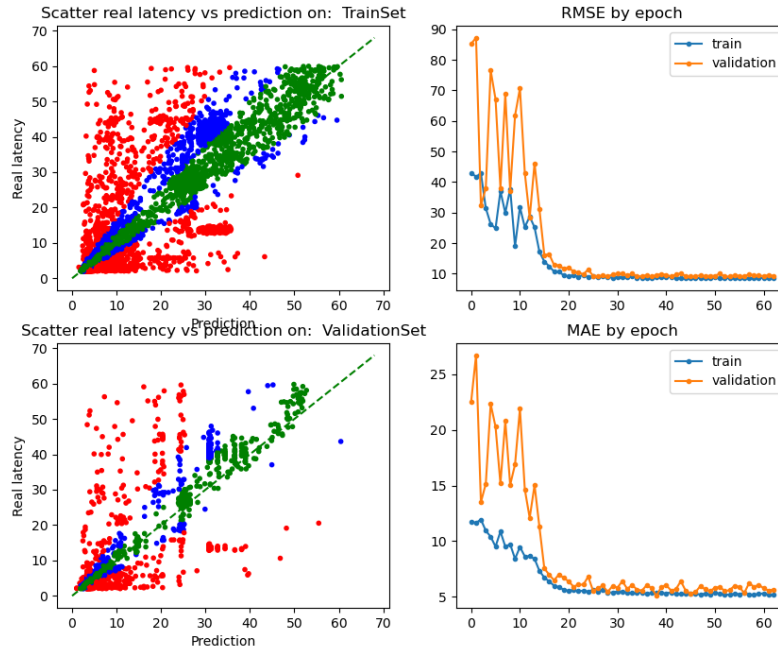


Figura 23: Entrenamiento en dataset combinado: DBPedia + LinkedGeoData + KEGG

Fuente: Elaboración Propia.

CAPÍTULO 5

CONCLUSIONES

Este trabajo de investigación se centró en la evaluación de un modelo de predicción de latencia para consultas SPARQL utilizando el conjunto de datos LSQ 2.0. A lo largo de este estudio, hemos explorado las complejidades inherentes a la predicción de tiempos de ejecución en bases de datos de grafos, específicamente en el contexto de consultas SPARQL sobre datos RDF. Los resultados obtenidos no solo validan la efectividad del modelo propuesto, sino que también abren nuevas vías de investigación y desarrollo en el campo de la optimización de consultas para bases de datos semánticas.

El modelo evaluado demostró una capacidad significativa para predecir los tiempos de ejecución de consultas SPARQL en diversos conjuntos de datos, incluyendo DBpedia, LinkedGeoData y KEGG. La utilización de técnicas de aprendizaje profundo, específicamente las convoluciones sobre árboles, permite capturar eficazmente la estructura compleja de las consultas SPARQL y sus planes de ejecución.

Sin embargo, es importante reconocer las limitaciones del estudio. El modelo actual, aunque

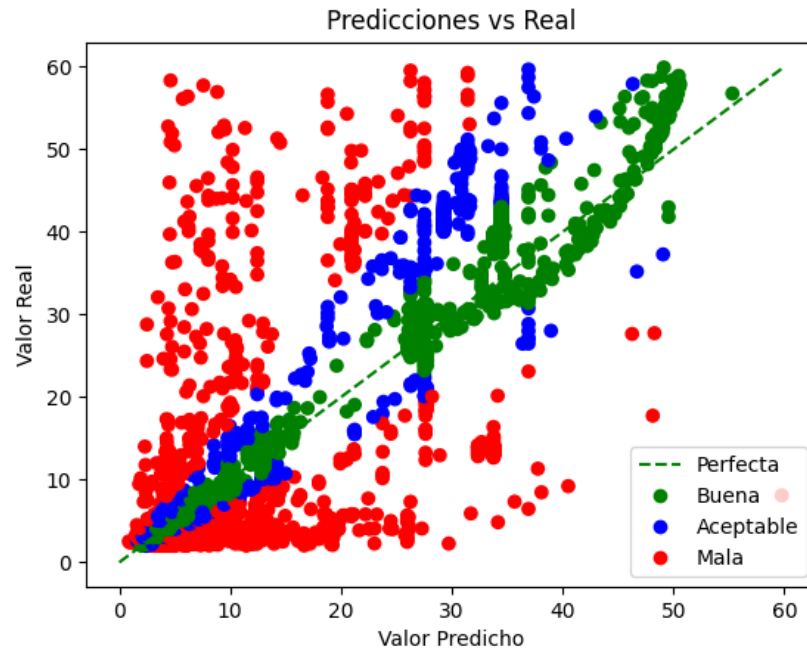


Figura 24: Predicciones en dataset combinado: DBPedia + LinkedGeoData + KEGG

efectivo, no considera toda la granularidad posible en la ejecución de consultas SPARQL. Aspectos como las ejecuciones de sub-consultas individuales dentro de una consulta compleja, cada una con su propia selectividad, tiempo y cantidad de resultados, aún no son consideradas como parte del modelo. Además, la variabilidad en los tiempos de ejecución observados en diferentes endpoints SPARQL sugiere que factores externos, como la carga del sistema y la optimización específica del motor de consultas, pueden influir significativamente en el rendimiento. También es importante destacar la poca capacidad del modelo para generalizar entre diferentes datasets, en especial aquellos que no tienen mucho en común, por lo que se considera una necesidad realizar adaptaciones específicas para cada endpoint.

Los resultados obtenidos muestran un desempeño más que aceptable en la precisión de la predicción de latencia. El uso de características a nivel de plan de ejecución demostró ser crucial para capturar la complejidad de las consultas SPARQL.

Revisando los objetivos planteados al inicio de esta investigación, podemos concluir que se han cumplido satisfactoriamente:

1. Se logró preparar y procesar eficazmente el conjunto de datos LSQ 2.0 para su uso en el modelo de predicción.
2. El modelo de predicción se implementó y ajustó con éxito para trabajar con los nuevos conjuntos de datos.

3. Se realizó una evaluación exhaustiva del rendimiento del modelo.
4. Los resultados se analizaron en profundidad, proporcionando información valiosa sobre la aplicabilidad del modelo en diferentes contextos de consultas SPARQL

5.1. Trabajo futuro

Basándonos en los resultados y las limitaciones identificadas, proponemos las siguientes líneas de investigación futura:

1. **Modelo mejorado con información granular:** Desarrollar un modelo que considere información más detallada sobre la ejecución de consultas SPARQL. Esto incluiría:
 - Modelar las distintas sub-ejecuciones necesarias para completar una consulta, cada una con su propia selectividad, tiempo y cantidad de resultados.
 - Incorporar características estáticas de las consultas presentes en LSQ que no fueron utilizadas en el modelo actual.
2. **Integración con Optimizadores de Consultas:** Dado el éxito del modelo actual y las posibles mejoras a realizar, un paso natural sería integrar el modelo de predicción dentro de un optimizador de consultas SPARQL, similar a lo realizado en [Marcus *et al.*, 2019].
3. **Adaptabilidad y Transferencia de Conocimiento:** Investigar métodos para hacer que el modelo sea más adaptable a diferentes endpoints SPARQL y conjuntos de datos.
4. **Técnicas avanzadas de balanceo de datasets:** Explorar y evaluar las técnicas de balanceo de datasets incorporadas en PyTorch, como WeightedRandomSampler o StratifiedSampler. Estas herramientas pueden proporcionar un enfoque más robusto y eficiente para manejar el desbalance en los tiempos de ejecución de las consultas, evitando la necesidad de realizar el balanceo de forma manual. Esto podría mejorar la capacidad del modelo para predecir con precisión tanto las consultas rápidas como las más lentas.
5. **Aplicación de técnicas de transfer learning:** Investigar y aplicar técnicas más recientes de transfer learning para mejorar la generalización del modelo a diferentes endpoints SPARQL y conjuntos de datos. Algunas opciones a considerar incluyen:
 - Fine-tuning adaptativo: Ajustar diferentes capas del modelo a diferentes tasas de aprendizaje para una mejor adaptación a nuevos dominios.
 - Meta-learning: Entrenar el modelo para que aprenda rápidamente de nuevos conjuntos de datos con pocas muestras.
 - Domain adaptation: Desarrollar técnicas para alinear las distribuciones de características entre diferentes endpoints SPARQL.

- Few-shot learning: Explorar métodos para adaptar el modelo a nuevos endpoints con un número limitado de ejemplos.

En conclusión, este trabajo reafirma el potencial de los modelos de aprendizaje profundo para predecir la latencia en consultas SPARQL, contribuyendo significativamente al avance de la Web Semántica y las tecnologías de bases de datos de grafos.

ANEXOS

Resumen resultados de predicciones

Modelo/Dataset	RMSE	MAE	R2	MAPE
Regular	9,318 $\pm$ 0,239	5,948 $\pm$ 0,407	0,661 $\pm$ 0,018	42,996 $\pm$ 2,023
Filtrado	10,526 $\pm$ 0,431	7,325 $\pm$ 0,405	0,590 $\pm$ 0,035	54,868 $\pm$ 3,287
Aum. 10 Bins, Lím. 30 %	9,334 $\pm$ 0,597	6,169 $\pm$ 0,474	0,658 $\pm$ 0,054	46,690 $\pm$ 3,324
Aum. 10 Bins, Lím. 40 %	9,439 $\pm$ 1,614	6,102 $\pm$ 0,483	0,642 $\pm$ 0,205	50,984 $\pm$ 4,467
Aum. 10 Bins, Lím. 50 %	9,424 $\pm$ 1,522	5,961 $\pm$ 0,438	0,644 $\pm$ 0,173	54,632 $\pm$ 4,597
Aum. 10 Bins, Lím. 60 %	9,542 $\pm$ 0,263	6,568 $\pm$ 0,317	0,644 $\pm$ 0,020	60,061 $\pm$ 4,484
Aum. 20 Bins, Lím. 30 %	9,198 $\pm$ 0,240	6,030 $\pm$ 0,329	0,669 $\pm$ 0,018	46,879 $\pm$ 3,244
Aum. 20 Bins, Lím. 40 %	9,400 $\pm$ 1,352	6,147 $\pm$ 0,642	0,648 $\pm$ 0,146	47,008 $\pm$ 4,978
Aum. 20 Bins, Lím. 50 %	9,297 $\pm$ 0,314	6,148 $\pm$ 0,410	0,662 $\pm$ 0,023	50,872 $\pm$ 3,083
Aum. 20 Bins, Lím. 60 %	9,232 $\pm$ 0,243	6,093 $\pm$ 0,396	0,667 $\pm$ 0,018	56,366 $\pm$ 3,611
Regular con características LSQ	9,422 $\pm$ 0,407	5,951 $\pm$ 0,357	0,631 $\pm$ 0,032	43,050 $\pm$ 2,231
Solo características LSQ	9,723 $\pm$ 0,264	6,314 $\pm$ 0,264	0,608 $\pm$ 0,024	47,214 $\pm$ 2,353
Solo características slice	9,648 $\pm$ 0,422	6,159 $\pm$ 0,391	0,613 $\pm$ 0,034	45,024 $\pm$ 1,744

Tabla 9: Métricas de evaluación de resultados en conjunto de test para DBPedia en distintos datasets y variaciones del modelo con nuevas características LSQ, los valores presentan Promedio $\pm$ Desviación Estándar de 100 ejecuciones

Modelo/Dataset	RMSE	MAE	R2	MAPE
Regular	20,345 $\pm$ 0,535	16,465 $\pm$ 0,367	-0,618 $\pm$ 0,086	215,248 $\pm$ 13,736
Aum. 10 Bins, Lím. 30 %	20,182 $\pm$ 0,5	16,421 $\pm$ 0,37	-0,592 $\pm$ 0,08	223,492 $\pm$ 11,589
Aum. 10 Bins, Lím. 40 %	20,142 $\pm$ 0,562	16,444 $\pm$ 0,416	-0,586 $\pm$ 0,088	229,93 $\pm$ 14,114
Aum. 10 Bins, Lím. 50 %	20,274 $\pm$ 0,488	16,593 $\pm$ 0,353	-0,607 $\pm$ 0,077	241,058 $\pm$ 11,989
Aum. 10 Bins, Lím. 60 %	19,947 $\pm$ 0,564	16,429 $\pm$ 0,449	-0,556 $\pm$ 0,088	239,665 $\pm$ 13,816
Aum. 20 Bins, Lím. 30 %	20,196 $\pm$ 0,524	16,428 $\pm$ 0,384	-0,594 $\pm$ 0,083	221,528 $\pm$ 13,155
Aum. 20 Bins, Lím. 40 %	20,395 $\pm$ 0,56	16,592 $\pm$ 0,421	-0,626 $\pm$ 0,091	228,073 $\pm$ 13,779
Aum. 20 Bins, Lím. 50 %	20,233 $\pm$ 0,566	16,535 $\pm$ 0,426	-0,6 $\pm$ 0,09	232,697 $\pm$ 14,187
Aum. 20 Bins, Lím. 60 %	20,176 $\pm$ 0,447	16,54 $\pm$ 0,337	-0,591 $\pm$ 0,071	241,854 $\pm$ 12,68

Tabla 10: Métricas de evaluación de resultados en conjunto de test para DBPedia en distintos datasets usando modelo NuSVR, los valores presentan Promedio $\pm$ Desviación Estándar de 100 ejecuciones

Modelo/Dataset	RMSE	MAE	R2	MAPE
Regular	10,279 $\pm$ 0,551	4,645 $\pm$ 0,327	0,273 $\pm$ 0,077	38,633 $\pm$ 3,573
Aum. 10 Bins, Lím. 30 %	9,981 $\pm$ 0,55	4,667 $\pm$ 0,35	0,314 $\pm$ 0,075	40,761 $\pm$ 4,161
Aum. 10 Bins, Lím. 40 %	9,45 $\pm$ 0,787	4,353 $\pm$ 0,388	0,385 $\pm$ 0,101	41,07 $\pm$ 4,093
Aum. 10 Bins, Lím. 50 %	9,782 $\pm$ 0,65	4,534 $\pm$ 0,362	0,341 $\pm$ 0,084	41,985 $\pm$ 3,61
Aum. 10 Bins, Lím. 60 %	9,511 $\pm$ 0,843	4,393 $\pm$ 0,469	0,377 $\pm$ 0,108	41,528 $\pm$ 4,36
Aum. 20 Bins, Lím. 30 %	9,844 $\pm$ 0,56	4,571 $\pm$ 0,307	0,333 $\pm$ 0,076	41,067 $\pm$ 2,947
Aum. 20 Bins, Lím. 40 %	9,888 $\pm$ 0,62	4,635 $\pm$ 0,377	0,327 $\pm$ 0,084	42,398 $\pm$ 4,153
Aum. 20 Bins, Lím. 50 %	9,681 $\pm$ 0,725	4,529 $\pm$ 0,418	0,355 $\pm$ 0,095	42,23 $\pm$ 4,418
Aum. 20 Bins, Lím. 60 %	9,657 $\pm$ 0,62	4,486 $\pm$ 0,378	0,358 $\pm$ 0,08	42,168 $\pm$ 4,22

Tabla 11: Métricas de evaluación de resultados en conjunto de test para LinkedGeoData en distintos datasets, los valores presentan Promedio $\pm$ Desviación Estándar de 100 ejecuciones

Resultados DBPedia

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	9,264	9,317	0,239	8,871	10,146
Filtrado	10,441	10,526	0.431	9,951	12,029
Aum. 10 Bins, Lím. 30 %	9,231	9,334	0,596	8,826	14,638
Aum. 10 Bins, Lím. 40 %	9,244	9,439	1,614	8,796	24,376
Aum. 10 Bins, Lím. 50 %	9,157	9,423	1,522	8,752	21,081
Aum. 10 Bins, Lím. 60 %	9,521	9,542	0,262	8,907	10,435
Aum. 20 Bins, Lím. 30 %	9,159	9,197	0,240	8,846	10,137
Aum. 20 Bins, Lím. 40 %	9,205	9,399	1,352	8,888	18,650
Aum. 20 Bins, Lím. 50 %	9,214	9,297	0,313	8,719	10,296
Aum. 20 Bins, Lím. 60 %	9,128	9,231	0,242	8,946	10,037
Regular con características LSQ	9,437	9,422	0,407	8,673	10,682
Solo características LSQ	9,780	9,723	0,301	8,979	10,243

Tabla 12: RMSE en conjunto de test para DBPedia, resultados para 100 ejecuciones

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	5,975	5,948	0,407	5,111	7,013
Filtrado	7,346	7,325	0,405	6,175	8,097
Aum. 10 Bins, Lím. 30 %	6,109	6,169	0,474	5,365	8,985
Aum. 10 Bins, Lím. 40 %	6,038	6,102	0,483	5,247	8,622
Aum. 10 Bins, Lím. 50 %	5,925	5,961	0,438	5,156	7,662
Aum. 10 Bins, Lím. 60 %	6,578	6,568	0,317	5,725	7,651
Aum. 20 Bins, Lím. 30 %	5,992	6,030	0,329	5,430	7,090
Aum. 20 Bins, Lím. 40 %	6,131	6,149	0,642	5,417	10,030
Aum. 20 Bins, Lím. 50 %	6,087	6,148	0,410	5,047	7,033
Aum. 20 Bins, Lím. 60 %	6,106	6,093	0,396	5,209	6,931
Regular con características LSQ	5,920	5,951	0,357	5,244	6,764
Solo características LSQ	6,324	6,314	0,264	5,620	6,730

Tabla 13: MAE en conjunto de test para DBPedia, resultados para 100 ejecuciones

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	0,665	0,661	0,018	0,598	0,693
Filtrado	0,597	0,590	0,035	0,465	0,634
Aum. 10 Bins, Lím. 30 %	0,667	0,658	0,054	0,163	0,696
Aum. 10 Bins, Lím. 40 %	0,666	0,642	0,205	-1,321	0,698
Aum. 10 Bins, Lím. 50 %	0,672	0,644	0,173	-0,736	0,701
Aum. 10 Bins, Lím. 60 %	0,646	0,644	0,020	0,575	0,690
Aum. 20 Bins, Lím. 30 %	0,672	0,669	0,018	0,599	0,694
Aum. 20 Bins, Lím. 40 %	0,669	0,648	0,146	-0,359	0,691
Aum. 20 Bins, Lím. 50 %	0,668	0,662	0,023	0,586	0,703
Aum. 20 Bins, Lím. 60 %	0,674	0,667	0,018	0,606	0,687
Regular con características LSQ	0,631	0,631	0,032	0,527	0,688
Solo características LSQ	0,603	0,608	0,024	0,565	0,666

Tabla 14: R2 en conjunto de test para DBPedia, resultados para 100 ejecuciones

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	42,750	42,996	2,023	38,875	49,629
Filtrado	54,442	54,868	3,287	49,548	67,860
Aum. 10 Bins, Lím. 30 %	46,265	46,690	3,324	41,174	63,876
Aum. 10 Bins, Lím. 40 %	49,932	50,984	4,467	44,218	82,691
Aum. 10 Bins, Lím. 50 %	53,800	54,632	4,597	47,208	78,576
Aum. 10 Bins, Lím. 60 %	59,350	60,061	4,484	52,152	75,592
Aum. 20 Bins, Lím. 30 %	46,598	46,879	3,244	41,528	61,533
Aum. 20 Bins, Lím. 40 %	46,510	47,008	4,978	41,984	77,059
Aum. 20 Bins, Lím. 50 %	50,270	50,872	3,083	46,690	64,857
Aum. 20 Bins, Lím. 60 %	56,225	56,366	3,611	50,505	64,906
Regular con características LSQ	42,465	43,050	2,231	39,424	47,449
Solo características LSQ	46,829	47,214	2,353	43,632	54,968

Tabla 15: MAPE en conjunto de test para DBPedia, resultados para 100 ejecuciones

Resultados LinkedGeoData

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	10,278	10,191	0,551	8,804	11,135
Aum. 10 Bins, Lím. 30 %	9.981	9,933	0,549	8,761	10,945
Aum. 10 Bins, Lím. 40 %	9.450	9,317	0,787	7,701	10,718
Aum. 10 Bins, Lím. 50 %	9.782	9,648	0,650	7,875	10,635
Aum. 10 Bins, Lím. 60 %	9.511	9,477	0,842	7,544	10,785
Aum. 20 Bins, Lím. 30 %	9.843	9,885	0,559	8,018	10,976
Aum. 20 Bins, Lím. 40 %	9.888	9,873	0,619	8,614	11,065
Aum. 20 Bins, Lím. 50 %	9.681	9,623	0,725	7,585	10,888
Aum. 20 Bins, Lím. 60 %	9.656	9,578	0,619	7,958	10,569

Tabla 16: RMSE en conjunto de test para LinkedGeoData

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	4.645	4.652	0.327	4.041	5.305
Aum. 10 Bins, Lím. 30 %	4.667	4.647	0.350	3.961	5.424
Aum. 10 Bins, Lím. 40 %	4.353	4.332	0.388	3.584	5.213
Aum. 10 Bins, Lím. 50 %	4.534	4.472	0.362	3.540	5.233
Aum. 10 Bins, Lím. 60 %	4.393	4.395	0.469	3.445	5.348
Aum. 20 Bins, Lím. 30 %	4.571	4.593	0.307	3.613	5.158
Aum. 20 Bins, Lím. 40 %	4.635	4.601	0.377	3.834	5.430
Aum. 20 Bins, Lím. 50 %	4.529	4.476	0.418	3.427	5.255
Aum. 20 Bins, Lím. 60 %	4.486	4.472	0.378	3.523	5.188

Tabla 17: MAE en conjunto de test para LinkedGeoData

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	0.273	0.283	0.077	0.147	0.467
Aum. 10 Bins, Lím. 30 %	0.314	0.319	0.075	0.176	0.472
Aum. 10 Bins, Lím. 40 %	0.385	0.398	0.101	0.209	0.592
Aum. 10 Bins, Lím. 50 %	0.341	0.356	0.084	0.221	0.573
Aum. 10 Bins, Lím. 60 %	0.377	0.377	0.108	0.199	0.608
Aum. 20 Bins, Lím. 30 %	0.333	0.325	0.076	0.171	0.558
Aum. 20 Bins, Lím. 40 %	0.327	0.326	0.084	0.157	0.489
Aum. 20 Bins, Lím. 50 %	0.355	0.359	0.095	0.184	0.604
Aum. 20 Bins, Lím. 60 %	0.358	0.366	0.080	0.231	0.564

Tabla 18: R2 en conjunto de test para LinkedGeoData

Dataset	Mediana	Promedio	Desv. Estándar	Mínimo	Máximo
Regular	38.633	39.066	3.573	32.517	50.160
Aum. 10 Bins, Lím. 30 %	40.761	41.505	4.161	33.380	51.707
Aum. 10 Bins, Lím. 40 %	41.070	41.397	4.093	33.751	54.928
Aum. 10 Bins, Lím. 50 %	41.985	41.833	3.610	36.232	52.870
Aum. 10 Bins, Lím. 60 %	41.528	41.296	4.360	34.547	53.338
Aum. 20 Bins, Lím. 30 %	41.067	41.075	2.947	34.169	49.318
Aum. 20 Bins, Lím. 40 %	42.398	42.351	4.153	32.008	54.254
Aum. 20 Bins, Lím. 50 %	42.230	42.094	4.418	34.150	53.769
Aum. 20 Bins, Lím. 60 %	42.168	42.915	4.220	33.927	52.749

Tabla 19: MAPE en conjunto de test para LinkedGeoData

REFERENCIAS BIBLIOGRÁFICAS

- [Akdere et al., 2012] Akdere, M., Çetintemel, U., Riondato, M., Upfal, E., y Zdonik, S. B. (2012). Learning-based query performance modeling and prediction. En *2012 IEEE 28th International Conference on Data Engineering*, pp. 390–401. IEEE.
- [Berners-Lee,] Berners-Lee, T. Linked data. <https://www.w3.org/DesignIssues/LinkedData.html>. [Online; Accedido el 15 de julio de 2024].
- [Casals et al., 2023] Casals, D., Buil-Aranda, C., y Valle, C. (2023). Sparql query execution time prediction using deep learning.
- [Duggan et al., 2011] Duggan, J., Cetintemel, U., Papaemmanouil, O., y Upfal, E. (2011). Performance prediction for concurrent database workloads. En *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 337–348.
- [Guus Schreiber,] Guus Schreiber, Y. R. Rdf 1.1 primer. <https://www.w3.org/TR/rdf11-primer/>. [Online; Accedido el 15 de julio de 2024].
- [Hasan y Gandon, 2014] Hasan, R. y Gandon, F. (2014). A machine learning approach to sparql query performance prediction. En *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volumen 1, pp. 266–273. IEEE.
- [Kossmann, 2000] Kossmann, D. (2000). The state of the art in distributed query processing. *ACM Computing Surveys (CSUR)*, 32(4):422–469.
- [Lassila, 1997] Lassila, O. (1997). Resource description framework (rdf) model and syntax.
- [Marcus et al., 2019] Marcus, R., Negi, P., Mao, H., Zhang, C., Alizadeh, M., Kraska, T., Papaemmanouil, O., y Tatbul, N. (2019). Neo: A learned query optimizer. *arXiv preprint arXiv:1904.03711*.
- [Miller, 1998] Miller, E. (1998). An introduction to the resource description framework. *??????????*, (13):3–11.
- [Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill series in computer science. McGraw-Hill Professional, New York, NY.
- [Mou et al., 2016] Mou, L., Li, G., Zhang, L., Wang, T., y Jin, Z. (2016). Convolutional neural networks over tree structures for programming language processing. En *Proceedings of the AAAI conference on artificial intelligence*, volumen 30.
- [Pan, 2009] Pan, J. Z. (2009). Resource description framework. En *Handbook on ontologies*, pp. 71–90. Springer.

- [Pavlo *et al.*, 2017] Pavlo, Andrew and Angulo, Gustavo and Arulraj, Joy and Lin, Haibin and Lin, Jiexi and Ma, Lin and Menon, Prashanth and Mowry, Todd C and Perron, Matthew and Quah, Ian and others (2017). Self-driving database management systems. En *CIDR*, volumen 4, p. 1.
- [Pérez *et al.*, 2010] Pérez, J., Arenas, M., y Gutierrez, C. (2010). nsparql: A navigational language for rdf. *Journal of Web Semantics*, 8(4):255–270.
- [Richard Cyganiak,] Richard Cyganiak, David Wood, M. L. Rdf 1.1 concepts and abstract syntax. <https://www.w3.org/TR/rdf11-concepts/>. [Online; Accedido el 15 de julio de 2024].
- [Stadler *et al.*, 2024] Stadler, C., Saleem, M., Mehmood, Q., Buil-Aranda, C., Dumontier, M., Hogan, A., y Ngonga Ngomo, A.-C. (2024). Lsq 2.0: A linked dataset of sparql query logs. *Semantic Web*, (Preprint):1–23.
- [Steve Harris, 2013] Steve Harris, Andy Seaborne, E. P. (2013). Sparql 1.1 query language. w3c recommendation. <https://www.w3.org/TR/sparql11-query/>. [Online; Accedido el 15 de julio de 2024].