Universidad de Valladolid

Máster universitario

Ingeniería Informática

Trabajo Fin de Máster

Evaluación de modelos generativos *on-premises* para la mejora de la calidad de datos tabulares

Realizado por **Michelle Valeria Plúas Vásquez**

✠✠✠

Universidad de Valladolid

18 de Julio de 2025

Tutor: Yania Crespo González-Carvajal

Universidad de Valladolid

Máster universitario en Ingeniería Informática

D. Yania Crespo González-Carvajal, profesora del departamento de Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que la estudiante D. Michelle Valeria Plúas Vásquez, ha realizado el Trabajo final de Máster en Ingeniería Informática titulado "Evaluación de modelos generativos *on-premises* para la mejora de la calidad de datos tabulares".

Y que dicho trabajo ha sido realizado por la estudiante bajo la dirección de quien suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Valladolid, 18 de Julio de 2025

Vº. Bº. del Tutor:

D. Yania Crespo González-Carvajal

Agradecimientos

Primero, gracias a Dios, por darme la fuerza para llegar hasta aquí.

A mi pareja, David, por estar presente durante todo el proceso. Por tu apoyo constante, por creer en mí y por ayudarme incluso cuando las cosas se pusieron cuesta arriba.

A mi tutora, Yania, por su acompañamiento a lo largo del desarrollo de este trabajo. Su disponibilidad, empatía y confianza en mí fueron clave para avanzar.

A Santiago, mi responsable directo, por su flexibilidad y comprensión durante esta etapa. Gracias por facilitarme continuar con el máster sin hacerme elegir entre el trabajo y el estudio.

A mis amigas Genesis y Thania, porque cuando todo me sobrepasaba, siempre encontraban la forma de distraerme. Me ayudaron más de lo que creen.

A mi familia, por estar. Incluso cuando no entendían bien qué estaba haciendo, me apoyaron igual.

Resumen

La limpieza de datos estructurados sigue siendo un proceso costoso y difícil de automatizar cuando los valores presentan errores, ausencias o inconsistencias. Recientes avances en modelos generativos han abierto la posibilidad de utilizar predicción de lenguaje para asistir estas tareas sin depender de reglas fijas o validaciones manuales.

Este trabajo comienza con una revisión del estado del arte sobre la aplicación de modelos generativos en calidad de datos. A partir de ese análisis, se adaptó un sistema base y se introdujeron errores controlados sobre un dataset clínico. Se evaluó el comportamiento de tres modelos locales ante datos con distintas alteraciones y se compararon sus salidas con los valores originales utilizando métricas por celda.

Los resultados muestran que el rendimiento varía según la variable analizada y que ciertos modelos ofrecen ventajas puntuales bajo condiciones específicas. Se observaron diferencias en precisión, recuperación y coincidencia exacta entre modelos y variables. Estas variaciones ayudan a entender cómo responde cada modelo ante errores estructurados y pueden orientar decisiones en tareas de limpieza específicas.

Descriptores

Calidad de datos, limpieza de datos, datos estructurados, modelos generativos, modelos de lenguaje, evaluación, recuperación de errores

Abstract

Structured data cleaning remains a costly process and difficult to automate when values contain errors, missing entries or inconsistencies. Recent advances in generative models have opened the possibility of using language prediction to assist these tasks without relying on fixed rules or manual validations.

This work begins with a review of the state of the art on the application of generative models to data quality. Based on this analysis, a base system was adapted and controlled errors were introduced into a clinical dataset. The behavior of three local models was evaluated against different alterations in the data, and their outputs were compared with the original values using cell-level metrics.

The results show that performance varies depending on the variable analyzed and that some models offer specific advantages under certain conditions. Differences were observed in precision, recall, and exact match across models and variables. These variations help to understand how each model responds to structured errors and can guide decisions in specific cleaning tasks.

Keywords

Data quality, data cleaning, structured data, generative models, language models, evaluation, error recovery

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto y Motivación

La calidad de los datos es un aspecto clave que impacta directamente en el funcionamiento diario y en la eficiencia global de las organizaciones. En muchos casos, los fallos que se atribuyen a procesos o servicios provienen en realidad de errores dentro del repositorio de datos, como registros duplicados, formatos inconsistentes o valores inválidos. Estas deficiencias generan costes innecesarios, dificultan la integración de información procedente de distintas fuentes y limitan el análisis posterior.Por eso, mantener buenos niveles de calidad se ha vuelto una prioridad. Normas técnicas como la ISO/IEC 8000 y la ISO/IEC 25012 reconocen que mantener altos niveles de calidad no solo evita pérdidas, sino que permite gestionar los datos como un activo estratégico, mediante procesos estructurados que garanticen su utilidad y fiabilidad a lo largo de todo su ciclo de vida.[22][14]

En entornos donde los datos circulan entre múltiples sistemas independientes, con estructuras y reglas propias, el control de calidad se vuelve más complejo. La heterogeneidad tecnológica y semántica, junto con la autonomía de cada componente, dificulta la aplicación de procedimientos uniformes para validar, limpiar o integrar la información. Esta falta de centralización y coordinación incrementa el riesgo de que errores iniciales se propaguen sin ser detectados, afectando la confiabilidad y utilidad de los datos a lo largo de todo su ciclo de vida. Por ello, es fundamental establecer mecanismos que permitan asegurar la calidad desde la fase de adquisición, limpieza y transformación, evitando así que problemas acumulativos generen consecuencias negativas para la toma de decisiones y la operativa empresarial[5].

Esto ha impulsado la búsqueda de alternativas más flexibles. Una de ellas es el uso de modelos generativos aplicados a tareas de mejora de calidad de datos. Estos modelos han sido explorados recientemente en contextos donde se requiere detectar errores, completar información o sugerir transformaciones, especialmente cuando los datos presentan ambigüedades o patrones irregulares. Modelos generativos como GPT-4, entrenados con grandes volúmenes de datos heterogéneos, han demostrado capacidad

para identificar errores, sugerir correcciones, completar información faltante o reformular contenido ambiguo, facilitando procesos de depuración de datos[3].

También se han propuesto enfoques que combinan procesamiento automatizado con interfaces de lenguaje natural, lo que permite que usuarios sin conocimientos técnicos puedan interactuar directamente con los sistemas de limpieza y transformación de datos. Estos métodos facilitan la generación de consultas o comandos de manipulación de datos mediante lenguaje cotidiano, simplificando tareas complejas como la depuración, transformación o análisis. Por ejemplo, herramientas recientes basadas en diferentes modelos de lenguaje de gran tamaño pueden interpretar consultas en lenguaje natural y generar automáticamente código para ejecutar operaciones específicas sobre grandes volúmenes de datos, incluso en entornos complejos como data lakes. Este tipo de solución reduce la dependencia de conocimientos especializados en programación o SQL, abriendo la puerta a un manejo más accesible y eficiente de los datos para un público más amplio[48].

Aunque estas líneas de trabajo aún están en desarrollo, ya existen estudios que apuntan a que se podría reducir la carga manual al facilitar la trazabilidad en los procesos de limpieza y transformación de datos. Además, estos enfoques basados en modelos generativos o sistemas que utilizan prompts reutilizables ofrecen una ventaja importante: la capacidad de adaptarse a distintos dominios sin requerir reglas predefinidas o configuraciones específicas para cada caso. Esto representa un avance respecto a los métodos tradicionales, que dependen en gran medida de un conocimiento experto para diseñar y mantener reglas de validación y transformación[33].

Sin embargo, su aplicación en escenarios reales plantea desafíos que van más allá de aspectos técnicos como la precisión o la reproducibilidad, involucrando también cuestiones críticas relacionadas con la protección de los datos, la seguridad y la privacidad.

Estas preocupaciones no se limitan únicamente a los usuarios finales, sino que afectan también a empresas, desarrolladores y organizaciones que integran estas plataformas en procesos productivos o que manejan información sensible. La naturaleza conversacional y humana de estas tecnologías puede incentivar la divulgación de datos personales o corporativos que, si no se gestionan adecuadamente, pueden derivar en filtraciones, pérdidas de propiedad intelectual o incumplimientos legales. Además, el uso de modelos alojados en la nube y gestionados por terceros introduce riesgos asociados a la falta de control directo sobre cómo se recopilan, almacenan y procesan los datos. Esto incluye la posibilidad de que la información sensible sea accesible para personal externo durante etapas de revisión o entrenamiento, o que se comparta con terceros sin el consentimiento explícito del usuario o la empresa.

Estos riesgos se intensifican en sectores con altos requisitos regulatorios, como finanzas, salud o servicios legales, donde la exposición o mal manejo de datos puede generar sanciones severas y daños reputacionales significativos[1]. Por eso, se plantea como alternativa el uso de modelos generativos locales, que puedan ejecutarse en infraestructuras controladas y minimizar la exposición de datos. Este enfoque es especialmente útil en entornos que priorizan la confidencialidad o trabajan bajo marcos normativos estrictos.

Este trabajo se alinea con esa dirección. La lectura del artículo *Can Generative AI Transform Data Quality*? [3], fue el punto de partida para realizar una revisión de literatura centrada en el uso de modelos generativos para tareas de limpieza de datos. En ese proceso se identificó RetClean, un sistema orientado a la depuración de datos tabulares que ya incluye, de forma limitada, un modelo generativo (*on-premises*). Al estar publicado como software de código abierto bajo licencia MIT [1], y contar con una arquitectura modular, permite incorporar nuevos modelos sin partir de cero[39]. El interés es comprobar si los modelos generativos, especialmente los ejecutados localmente, pueden complementar los enfoques actuales de limpieza de forma práctica y sostenible.

1.2. Objetivos

Después de haber expuesto el planteamiento, el contexto y la motivación, se expone el objetivo principal de este TFM:

- Aplicar y evaluar modelos de lenguaje de gran tamaño en tareas de mejora de calidad de datos estructurados.

Con el fin de desglosar el objetivo principal, se definen los siguientes objetivos secundarios:

- Revisar la literatura científica y técnica sobre el uso de modelos de lenguaje de gran tamaño en tareas de limpieza y preparación de datos.
- Analizar el comportamiento del sistema base elegido a partir de la revisión de la literatura, en este caso RetClean, incluyendo su funcionamiento actual y las limitaciones que presenta en tareas de limpieza.
- Identificar posibles los riesgos de privacidad y protección de datos asociados al uso de modelos generativos.
- Integrar y adaptar modelos generativos dentro del flujo de trabajo de limpieza de datos definido por el sistema base elegido, identificando los puntos del sistema que requieren ajustes para su correcta incorporación.
- Evaluar su capacidad para identificar, corregir o sugerir mejoras en datos tabulares.
- Comparar los resultados obtenidos frente a los datos originales, considerando precisión, esfuerzo requerido y tipo de errores tratados.

[1] https://github.com/qcri/RetClean/blob/main/LICENSE

1.3. Preguntas de investigación

En este sentido, para esclarecer más sobre este tema se han identificado algunas preguntas de investigación:

RQ1 : ¿Qué enfoques recientes han explorado el uso de modelos generativos en tareas de limpieza y mejora de calidad de datos estructurados?

RQ2 : ¿Qué tareas de limpieza pueden ser asistidas eficazmente por modelos de lenguaje generativos?

RQ3 : ¿Qué ventajas o limitaciones presentan los LLMs frente a técnicas tradicionales de limpieza?

RQ4 : ¿En qué medida los modelos seleccionados permiten adaptaciones para distintos contextos de calidad de datos?

RQ5: ¿Qué riesgos de privacidad y protección de datos pueden surgir al utilizar modelos generativos en sistemas locales de limpieza de datos?

1.4. Estructura del documento

El presente documento se organiza de la siguiente manera:

- **Capítulo 1: Introducción.** Presenta el contexto del trabajo, los objetivos y las preguntas de investigación que guían el estudio.
- **Capítulo 2: Marco teórico.** Revisa los conceptos clave sobre sobre calidad de datos, técnicas de limpieza automatizada, modelos generativos, herramientas de procesamiento de datos estructurados e interfaces de lenguaje natural y así como las métricas empleadas para evaluar sus respuestas.
- **Capítulo 3: Estado del arte.** Presenta el resultado de una revisión sistemática de la literatura sobre el uso de modelos de lenguaje en tareas de mejora de calidad de datos, distinguiendo entre propuestas metodológicas, herramientas y casos de aplicación.
- **Capítulo 4: Caso de estudio.** Expone cómo se utilizó la herramienta RetClean, el conjunto de datos, la preparación del entorno, los pasos realizados y los criterios de evaluación.
- **Capítulo 5: Resultados y Discusión.** Presenta los resultados cuantitativos y ejemplos obtenidos. Se analizan los comportamientos observados en los modelos y se discuten sus ventajas, limitaciones y condiciones de uso.

- **Capítulo 6: Conclusiones.** Resume las principales conclusiones del estudio y plantea posibles líneas futuras de trabajo.
- **Apéndice A: Plan de proyecto.** Incluye las herramientas utilizadas, la metodología de trabajo, el plan inicial y el seguimiento de la planificación.
- **Apéndice B: Requisitos de hardware.** Recoge los componentes técnicos mínimos necesarios para ejecutar el sistema, incluyendo CPU, GPU, memoria y almacenamiento.
- **Apéndice C: Guía técnica de instalación y uso.** Describe los pasos necesarios para ejecutar el sistema y reproducir los experimentos.
- **Apéndice D: Estructura del repositorio.** Presenta la organización de carpetas del sistema base adaptado, con una breve descripción funcional de cada directorio y su rol dentro del flujo de experimentación.

Capítulo 2

Marco teórico

Este capítulo recoge las bases teóricas del trabajo. Se define qué es calidad de datos y cuáles son sus dimensiones. Se explican las técnicas de limpieza más usadas y los problemas frecuentes en datos estructurados. Se introduce el funcionamiento general de los modelos generativos y de lenguaje, y su aplicación en tareas de mejora de datos. Se describe cómo se integran en pipelines automatizados y cómo se accede a ellos mediante lenguaje natural. También se presentan las métricas de evaluación aplicables a tareas de generación, restauración y comparación de datos.

2.1. Calidad de datos

Definición y dimensiones de la calidad de datos

Para definir la calidad de los datos, primero hay que establecer qué se entiende por dato. La definición varía según el autor y el contexto; por ejemplo, Fu y Easton [23] lo definen como el resultado de registrar hechos del mundo real mediante observación o medición. Un dato puede almacenarse, contener información útil y ser interpretado por un usuario. En teoría, debería representar un hecho verdadero, pero en la práctica esto no siempre ocurre. Errores en la captura, el almacenamiento o el procesamiento pueden afectar su fidelidad desde el origen.

Dado que los datos no siempre reflejan con precisión la realidad que intentan representar, surge la necesidad de evaluar su calidad. La calidad de los datos se entiende como el conjunto de características que determinan en qué medida los datos cumplen con los requisitos específicos del usuario en un contexto determinado[8]. Estas características se expresan en dimensiones. Una calidad deficiente puede comprometer la eficacia de los sistemas que utilizan dichos datos. La evaluación de estas dimensiones depende del tipo de tarea, del dominio de aplicación y de las expectativas del usuario final. Por tanto, la calidad de los datos no puede evaluarse de forma aislada, sino en función de su uso[23].

Dimensión	**Descripción**	**Ejemplo**
Precisión	Mide qué tan cerca está el valor registrado del valor real.	Una dirección es precisa si corresponde a la ubicación física del cliente.
Completitud	Indica si todos los datos requeridos están presentes.	Si falta el número de teléfono en un registro obligatorio, la completitud es baja.
Consistencia	Verifica que no existan contradicciones entre registros o fuentes.	Si la fecha de nacimiento de una persona difiere entre sistemas, hay inconsistencia.
Actualidad	Evalúa si los datos reflejan el estado más reciente del objeto que describen.	Un inventario debe mostrar la disponibilidad real del producto.
Validez	Revisa si los datos cumplen con el formato, tipo o rango esperado.	Un campo que solo acepta valores entre 1 y 100 es inválido si contiene un 105.
Unicidad	Garantiza que no existan duplicados de una misma entidad.	Dos registros idénticos para el mismo cliente indican baja unicidad.
Relevancia	Determina si los datos son útiles para el propósito previsto.	Un dato es relevante si contribuye a la toma de decisiones en un proceso específico.
Accesibilidad	Mide la facilidad con la que los usuarios autorizados acceden a los datos cuando los necesitan.	Los usuarios deben poder acceder a los datos sin barreras técnicas o permisos innecesarios.
Interpretabilidad	Evalúa si los datos pueden ser comprendidos con facilidad por los usuarios.	Un campo etiquetado claramente facilita la interpretación del contenido.

Tabla 2.1: Dimensiones de la calidad de los datos según Carlo Batini y Monica Scannapieca

Además de las dimensiones habitualmente mencionadas en la literatura, existen estándares internacionales que proporcionan un marco formal para evaluar la calidad de los datos. Uno de los más reconocidos es la norma ISO/IEC 25012[27], que forma parte de la familia de estándares SQuaRE (Software Product Quality Requirements and Evaluation). Esta norma define un modelo de calidad de datos que distingue entre dos grandes tipos de características: inherentes y dependientes del sistema.

Las **características inherentes** hacen referencia a propiedades propias del dato, independientemente de su entorno tecnológico. Estas incluyen:

- **Exactitud (Accuracy)**: grado en que los datos representan correctamente los valores del mundo real.
- **Completitud (Completeness)**: medida en que los datos están presentes en su totalidad.
- **Consistencia**: ausencia de contradicciones en los datos.

- **Credibilidad (Credibility)**: grado de confianza que se puede depositar en los datos.
- **Actualidad (Currentness)**: grado en que los datos están actualizados.

Por otro lado, las **características dependientes del sistema** dependen del contexto tecnológico en el que los datos se gestionan, y comprenden:

- **Accesibilidad**: grado en que los datos pueden ser recuperados por los usuarios autorizados.
- **Portabilidad**: facilidad con que los datos pueden ser transferidos entre entornos.
- **Seguridad**: protección contra accesos no autorizados o modificaciones indebidas.
- **Trazabilidad**: capacidad para identificar el origen y el historial de los datos.
- **Comprensibilidad (Understandability)**: facilidad con que los usuarios pueden interpretar los datos.

Este modelo estandarizado[27] proporciona un marco más formal y aplicable a entornos empresariales y tecnológicos, y permite alinear las evaluaciones de calidad con buenas prácticas reconocidas internacionalmente.

Ciclo de vida del dato y sus puntos críticos

La calidad del dato no es un estado estático ni garantizado. A lo largo del ciclo que va desde la observación del mundo real hasta el uso de los datos en sistemas, pueden introducirse errores, ambigüedades y pérdidas que afectan su valor informativo. Como muestra la Figura 2.1, el deterioro puede originarse en múltiples fases: desde la percepción inicial hasta el diseño de las estructuras, el almacenamiento o el uso final [8].

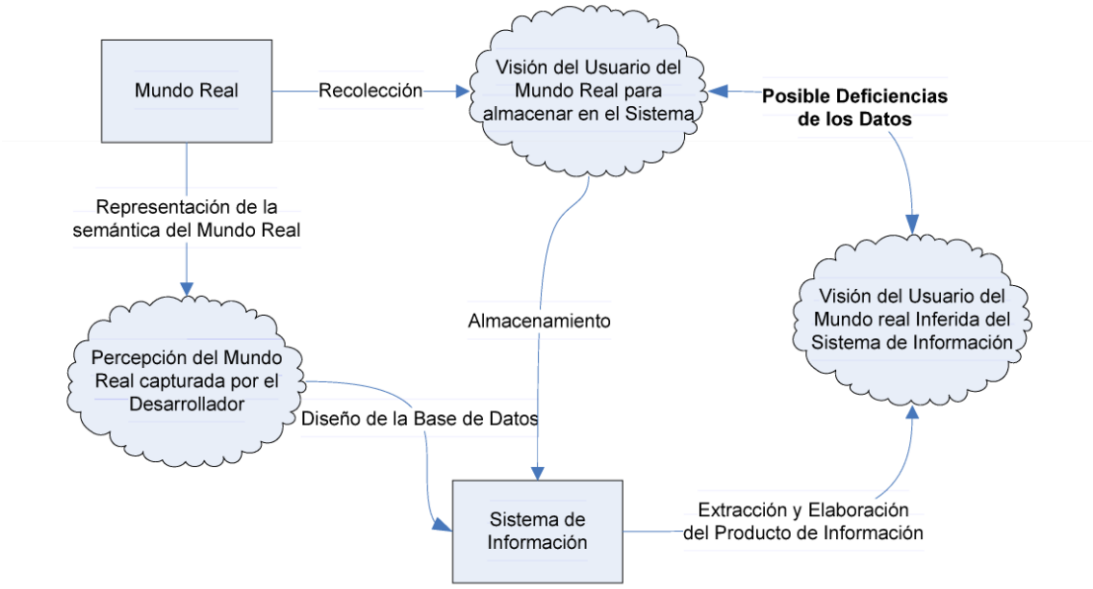

Figura 2.1: Fuentes de problemas de calidad de datos. Adaptado de Wand y Wang (1996)

Para entender y gestionar este deterioro, McGilvray propone el modelo POSMAD, un marco que identifica seis etapas fundamentales. A continuación, se describe cada fase junto con las actividades típicas y los principales puntos críticos que pueden comprometer la calidad [38].

1. **Planificar**: consiste en definir los objetivos, estándares, modelos de datos y estructuras necesarias antes de que los datos entren en producción. Aquí se diseña la arquitectura y se establecen las definiciones que guiarán el uso posterior. Si esta fase se omite o se ejecuta sin alinear diseño y uso real, los errores se trasladan al resto del ciclo.

2. **Obtener**: implica la adquisición de datos desde fuentes internas o externas. Las actividades incluyen la creación de registros, la carga de archivos o la compra de datos. Los riesgos en esta etapa incluyen entradas incorrectas, falta de validaciones o el uso de fuentes poco confiables.

3. **Almacenar y compartir**: en esta etapa los datos se guardan en sistemas y se ponen a disposición de otros procesos o usuarios. Pueden almacenarse en bases de datos o ficheros y compartirse a través de redes, pantallas o informes. Los puntos críticos incluyen duplicados, obsolescencia, problemas de integridad o controles de acceso mal definidos.

4. **Mantener**: esta fase abarca las operaciones de actualización, limpieza, transformación y consolidación de los datos. Aquí se aplican reglas de calidad, se corrigen

errores y se eliminan redundancias. Un mal mantenimiento puede generar pérdida de información válida o introducir errores adicionales.

5. **Aplicar**: los datos se utilizan para tomar decisiones, generar informes, entrenar modelos o ejecutar procesos automáticos. Si los datos aplicados son incompletos, incorrectos o mal interpretados, los errores se trasladan directamente a las decisiones o resultados del sistema.

6. **Disponer**: cuando los datos dejan de ser útiles, deben archivarse o eliminarse según las políticas definidas. Si se eliminan datos relevantes antes de tiempo, o si se conservan registros innecesarios sin contexto, se afecta tanto la trazabilidad como el cumplimiento normativo.

Estas fases se superponen con las causas clásicas de degradación de calidad. El deterioro puede clasificarse en tres categorías: operacional, conceptual y organizacional. La Tabla 2.2 resume los síntomas y causas más comunes [8].

Categoría	**Problemas**	**Causas**
Operacional	Los datos no existen, no son precisos o no son válidos.	Fallos de captura o transmisión.
Conceptual	Los datos están perdidos, no son precisos o no son válidos.	No están bien definidos o no son aptos para el uso previsto.
Organizacional	Hay problemas operacionales o conceptuales que se repiten.	Desconexión entre quien recoge los datos y quien los usa.

Tabla 2.2: Descripción de problemas y causas por categoría

Integrar una perspectiva de ciclo de vida, como la que ofrece POSMAD, permite identificar en qué fase se generan los problemas y qué actores están implicados.

2.2. Limpieza de datos

Concepto y objetivos

La limpieza de datos [36], también llamada data cleaning, consiste en detectar y corregir errores que distorsionan la representación del mundo real. Su objetivo es mejorar la calidad de los datos mediante la eliminación de anomalías, es decir, valores que introducen inconsistencias o imprecisiones en el conjunto. No hay una definición única sobre el alcance de este proceso, ya que varía según el dominio y los requisitos específicos del sistema.

Todo enfoque de limpieza debe cumplir al menos tres condiciones. Primero, debe permitir detectar y corregir errores tanto en fuentes aisladas como al integrar datos de distintos orígenes. Segundo, debe apoyarse en herramientas que minimicen la inspección manual y el desarrollo a medida, y que puedan adaptarse a nuevas fuentes. Tercero, debe coordinarse con las transformaciones de datos basadas en esquemas y metadatos [45].

Las anomalías que busca corregir este proceso pueden clasificarse, según Batini[4], en:

- **Sintácticas:** errores de formato, dominios no válidos o datos sin estandarizar.
- **Semánticas:** violaciones de integridad, duplicados o registros que contradicen otras dependencias del conjunto.
- **De contexto:** valores o tuplas que faltan, pero deberían existir según el modelo del dominio.

Una de las anomalías más frecuentes definidas por Beatriz Lopez[36] en el libro *Limpieza de datos* es la ausencia de valor. Puede deberse a errores humanos, fallos en sistemas de adquisición o a situaciones donde el valor no aplica. Se distingue entre:

- **Dato ausente:** existe en el mundo real, pero no fue registrado.
- **Dato nulo:** no tiene un valor definido o aplicable dentro del dominio del atributo.

Técnicas tradicionales

Las técnicas tradicionales han constituido la base de la gestión de la calidad de los datos durante décadas y siguen siendo esenciales para establecer entornos de datos fiables [5]. Se apoyan en enfoques manuales o en la aplicación de reglas definidas y procedimientos estadísticos [26]:

- La limpieza manual, implica la revisión directa de los datos por parte de un humano. Aunque es costosa en términos de tiempo y recursos, resulta necesaria en situaciones donde se requiere interpretación contextual o conocimiento experto. Este enfoque suele incluir inspecciones visuales, corrección directa de valores y resolución de duplicidades mediante revisión asistida.
- La limpieza basada en reglas y scripts, ofrece un enfoque más escalable. Consiste en definir condiciones lógicas o reglas de negocio que se implementan mediante scripts de programación o herramientas de procesamiento de datos. Este método permite verificar que los valores cumplen con el formato y dominio esperado, normalizar entradas con distintas representaciones, detectar desviaciones estadísticas, rellenar o eliminar valores faltantes según condiciones predefinidas, identificar duplicidades mediante técnicas de coincidencia aproximada y validar relaciones entre atributos dependientes.
- Las técnicas estadísticas y algorítmicas, complementan los enfoques anteriores mediante métodos de análisis de patrones. A través del estudio de la distribución y frecuencia de los valores, permiten detectar irregularidades que podrían indicar errores. Además, utilizan modelos para predecir datos faltantes o corregir inconsistencias que no siguen el comportamiento general del conjunto. Aunque estos métodos

requieren mayor capacidad computacional y conocimiento técnico, son útiles cuando se trabaja con grandes volúmenes de datos o cuando las inconsistencias no pueden definirse con reglas simples.

Retos en datos estructurados

Los datos estructurados se almacenan en formatos predefinidos, como tablas con filas y columnas. Aunque esta organización permite su procesamiento sistemático, no elimina los riesgos de errores ni garantiza su calidad. Al contrario, los datos estructurados presentan retos específicos que requieren atención continua y mecanismos adecuados para su limpieza [5].

Uno de los principales problemas son las inconsistencias y duplicidades a gran escala. Es común que una misma entidad esté representada de forma diferente en múltiples registros. Esto se agrava en procesos de integración de fuentes diversas, como sistemas de clientes o inventario. Detectar y fusionar estos casos requiere técnicas avanzadas de desduplicación [10].

También es habitual encontrar valores faltantes o atípicos. Ausencias sistemáticas o puntuales pueden tener origen en fallos de captura, errores de sistema o condiciones reales no previstas. Por su parte, los valores extremos deben analizarse para determinar si son errores o casos válidos, lo que exige conocimiento del dominio y métodos adecuados para tratarlos [26].

Otro reto importante son los errores semánticos. Aunque los formatos sean correctos, los datos pueden tener significados diferentes según la fuente. Esto ocurre, por ejemplo, cuando se mezclan unidades de medida o se aplican definiciones distintas a un mismo atributo. La integración heterogénea de sistemas como CRM o ERP suele acentuar estas inconsistencias [5, 10].

2.3. Procesamiento automatizado de datos

Principios del procesamiento automatizado de datos

El procesamiento automatizado de datos consiste en ejecutar tareas de manipulación, transformación, análisis y gestión de datos mediante sistemas informáticos sin intervención humana o con una intervención mínima. Su finalidad es realizar estas operaciones de manera eficiente, repetible y a gran escala, superando las limitaciones del trabajo manual en términos de velocidad y precisión. Estos sistemas operan de forma autónoma una vez configurados y abarcan desde la captura inicial y la limpieza de datos hasta la generación de informes o el suministro de información a otros sistemas [55].

A diferencia del procesamiento manual, que es lento y propenso a errores, y del procesamiento asistido, que todavía requiere supervisión, la automatización ofrece mayor velocidad, una precisión constante y la capacidad de manejar volúmenes masivos con

supervisión mínima. Su principal valor está en la capacidad de repetir procesos sin desviaciones y liberar recursos humanos para tareas de mayor valor estratégico [26].

La automatización es un componente esencial del enfoque orientado por datos. Para que una organización base sus decisiones en información actual y confiable, debe ser capaz de procesar grandes cantidades de datos con rapidez. Este enfoque permite que la información esté disponible para el análisis continuo sin depender de procesamiento manual, haciendo posible operar en tiempo real [55].

Arquitectura de data lakes

Los data lakes [25] representan una arquitectura diseñada para almacenar grandes volúmenes de datos sin importar su estructura. Su base teórica se apoya en la necesidad de escalabilidad y flexibilidad. Al mantener los datos en su forma original, se retrasa la definición del esquema hasta el momento en que se consultan o procesan. Este enfoque, conocido como schema-on-read, permite reutilizar los datos en distintos contextos y aplicar técnicas de análisis avanzadas sin restricciones previas. La diferencia principal entre un data lake y un data warehouse es precisamente la forma en que se estructuran los datos. En un data warehouse se sigue el enfoque schema-on-write, donde los datos deben ajustarse a una estructura definida antes de su almacenamiento. Esta distinción convierte a los data lakes en una solución adecuada para flujos de trabajo automatizados, donde se requiere una ingesta continua de información desde múltiples fuentes. Funcionan como un punto de entrada eficiente para procesos de limpieza, transformación y análisis, permitiendo una gestión integral de los datos a cualquier escala.

2.4. Modelos generativos y LLMs

Introducción a los modelos generativos

La inteligencia artificial generativa es un campo que se desarrolla a partir del aprendizaje automático. Su principio fundamental es la capacidad de un sistema para crear contenido nuevo a partir de patrones aprendidos. Estos sistemas no requieren reglas programadas ni datos etiquetados. En su lugar, analizan grandes volúmenes de información y construyen representaciones internas que permiten generar texto, imágenes o audio con estructura coherente. La generación se basa en procesos estadísticos y optimización de resultados a través de retroalimentación. Este enfoque transforma la lógica tradicional de la IA, al desplazar el foco desde la ejecución de instrucciones hacia la creación autónoma de contenido [2].

Los modelos generativos implementan este principio al modelar directamente la distribución de los datos. Su objetivo no es clasificar ni seleccionar, sino construir nuevas instancias que conserven la estructura del conjunto original. Aprenden a predecir secuencias de elementos, ajustando sus parámetros para maximizar la coherencia. El entrenamiento de estos modelos implica aprender correlaciones, dependencias y regularidades dentro de

los datos. Su funcionamiento se diferencia del de los modelos discriminativos, que solo se enfocan en asignar etiquetas a entradas. Mientras los discriminativos definen límites, los generativos definen el espacio completo. Los modelos generativos se clasifican según su estructura y la forma en que aprenden la distribución de los datos. La Tabla 2.3 resume los principales tipos utilizados en sistemas actuales [6].

Tipo de modelo	**Propósito**	**Base técnica**	**Descripción**
Autoencoder (AE)	Reconstrucción de la entrada	Codificación y decodificación determinista	Modelo no generativo en sí. Aprende a comprimir y descomprimir datos para reconstruir la entrada original.
Variational Autoencoder (VAE)	Generación de nuevas instancias probabilísticas	Inferencia bayesiana con muestreo estadístico	Variante generativa del autoencoder. Aprende distribuciones latentes y genera datos similares a los de entrenamiento.
Transformer	Procesamiento de secuencias complejas	Mecanismo de atención y codificación profunda	Modelo secuencial que aprende relaciones a largo alcance. Base para tareas de traducción, síntesis y resumen.
Generative Adversarial Network (GAN)	Síntesis de datos realistas	Competencia entre generador y discriminador	Dos redes compiten para generar datos indistinguibles del original. Útil en imagen, video y datos sintéticos.

Tabla 2.3: Tipos de modelos generativos basada en Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers

Modelos de lenguaje de gran tamaño

Los modelos de lenguaje de gran tamaño (*Large Language Models* en inglés), también llamados LLMs por sus siglas en inglés, son una aplicación directa de los modelos generativos al procesamiento de texto. Están diseñados para predecir tokens, que son unidades mínimas de texto como palabras o fragmentos, dentro de una secuencia, aprendiendo el contexto y las relaciones internas del lenguaje. La escala y la arquitectura los distinguen. Estos modelos operan con estructuras neuronales profundas, entrenadas sobre grandes corpus de texto. Su objetivo es generar lenguaje natural de forma autónoma, manteniendo coherencia gramatical, semántica y contextual. La técnica de aprendizaje autosupervisado les permite ajustarse a múltiples tareas con un solo sistema base [37].

Entre los LLMs más utilizados se encuentran [2]:

- **BERT**. Bidirectional Encoder Representations from Transformers. Utiliza arquitectura Transformer. Es un modelo preentrenado que aplica aprendizaje autosupervisado mediante la predicción de palabras enmascaradas dentro de secuencias de texto. Su funcionamiento bidireccional le permite captar el contexto desde ambos lados de

una palabra. Aprende representaciones contextuales profundas y puede ajustarse a tareas específicas como clasificación, pregunta-respuesta y traducción [16].

- **GPT-4**. Generative Pretrained Transformer versión 4. Utiliza arquitectura Transformer multimodal. Acepta entradas de texto e imagen y genera salidas de texto. Está preentrenado para predecir tokens y posteriormente alineado mediante técnicas de aprendizaje por refuerzo con retroalimentación humana. Mejora la factualidad y el comportamiento respecto a versiones anteriores [41].

- **LaMDA**. Language Model for Dialogue Applications. Utiliza arquitectura Transformer. Está entrenado específicamente para el diálogo, con un corpus diseñado para cubrir conversaciones abiertas. Su funcionamiento se centra en captar matices conversacionales y mantener la coherencia a lo largo del intercambio [49].

- **LLaMA**. Large Language Model Meta AI. Utiliza arquitectura Transformer autorregresiva. Está diseñado para ofrecer rendimiento competitivo con una menor cantidad de parámetros. Se entrena con grandes volúmenes de texto y predice tokens de manera secuencial. Mantiene eficiencia sin sacrificar calidad en la generación [50].

- **BLOOM**. BigScience Large Open-science Open-access Multilingual Language Model. Utiliza arquitectura Transformer. Se entrena con aprendizaje no supervisado para generar texto coherente y fluido en múltiples lenguas. Su funcionamiento permite capturar estructuras lingüísticas complejas y relaciones semánticas profundas [2].

- **Mistral**. Modelo de lenguaje extenso desarrollado por Mistral AI. Utiliza arquitectura Transformer. La versión Mistral 7B está compuesta por 7 mil millones de parámetros. Incorpora mecanismos de atención agrupada y atención en ventana deslizante que permiten mantener la eficiencia en el procesamiento secuencial. Está diseñado para tareas de generación de texto, razonamiento estructurado y comprensión multilingüe [28].

- **LLaVA**. Modelo multimodal que combina un codificador visual con un modelo de lenguaje. Basado en arquitectura Transformer. Está entrenado de forma conjunta con datos textuales e instrucciones visuales. Permite interpretar imágenes junto con texto y generar respuestas alineadas con ambos tipos de entrada. Está diseñado para tareas de diálogo con entradas visuales [35].

RAG: Métodos de recuperación y generación

La técnica de *Retrieval-Augmented Generation* (RAG) combina la potencia de los modelos de lenguaje con la precisión de fuentes de información externas, mejorando así la calidad y fiabilidad de las respuestas generadas. En un sistema RAG, cuando un usuario realiza una consulta, el modelo primero recupera información relevante de una base de datos externa. Posteriormente, esta información se integra con la consulta original y se envía a un modelo de lenguaje grande (LLM), que genera una respuesta coherente y precisa, fundamentada en datos actualizados y específicos del contexto [15].

Este enfoque es especialmente útil en dominios donde la precisión y la actualización de la información son críticas, como en el ámbito legal, científico o administrativo. Al incorporar datos externos en tiempo real, RAG permite que los modelos de lenguaje generen respuestas más pertinentes y menos propensas a errores o "alucinaciones", mejorando así la confianza y utilidad de las aplicaciones de inteligencia artificial generativa [15].

2.5. Interfaces de lenguaje natural

Una interfaz de lenguaje natural [30] permite que el usuario se comunique con un sistema utilizando expresiones propias del lenguaje humano. Sustituye las interfaces tradicionales compuestas por botones, menús o sintaxis técnicas. En su lugar, el usuario introduce instrucciones mediante texto, voz o signos, y el sistema responde utilizando el mismo canal. En el caso de los sistemas que procesan datos automáticamente, estas interfaces reducen la barrera técnica al eliminar la necesidad de conocer comandos estructurados como SQL o expresiones lógicas complejas.

El funcionamiento se basa en una serie de componentes que procesan el lenguaje del usuario. Entre ellos se encuentran el análisis léxico para identificar unidades del texto, el análisis sintáctico para determinar la estructura gramatical, y el análisis semántico para interpretar el significado. En muchos casos también se aplica análisis morfológico, extracción de entidades o identificación de intención. Estos procesos permiten traducir una consulta en lenguaje natural a una instrucción formal que puede ser ejecutada por el sistema.

Las tareas que se pueden realizar a través de interfaces de lenguaje natural incluyen la recuperación de datos, la ejecución de operaciones, la navegación por estructuras complejas y la generación automática de contenido. En el campo de la generación de lenguaje natural, estas tareas se clasifican según el tipo de entrada en texto a texto, datos a texto, imagen a texto y video a texto. Cada una de ellas requiere métodos distintos. Por ejemplo, la generación de texto a partir de datos puede realizarse con plantillas predefinidas o mediante modelos entrenados con grandes corpus. Las salidas pueden incluir resúmenes, traducciones, respuestas o reformulaciones de información.

En el marco de las interfaces de lenguaje natural, un prompt [24] es una entrada específica que se le proporciona a un modelo de inteligencia artificial para generar una respuesta acorde al objetivo deseado. Actúa como punto de partida del proceso de generación. Su estructura define el tipo de salida que el sistema puede producir. Un prompt puede adoptar la forma de una pregunta, una instrucción directa o una descripción de contexto. La precisión del resultado depende directamente de la claridad del prompt. Por ejemplo, una petición genérica genera una respuesta amplia, mientras que una instrucción detallada permite obtener información más precisa y ajustada. En aplicaciones avanzadas, un prompt puede incluir la simulación de un rol o estilo específico para guiar el tono y el enfoque de la respuesta. El diseño de prompts efectivos se ha convertido en una práctica clave para mejorar el rendimiento de los modelos generativos, y su elaboración, según Vladimir Geroimenko[24], se conoce como ingeniería de prompts.

2.6. Evaluación y métricas

Métricas de evaluación de la calidad en tareas de emparejamiento de datos

La evaluación en tareas de emparejamiento de datos [12] consiste en medir cuán correctamente un sistema identifica registros que se refieren a la misma entidad. Para ello se comparan los resultados del sistema con un conjunto de referencia conocido como *ground-truth*, que contiene el estado real de cada par. Este conjunto puede generarse a partir de revisión manual, resultados previamente validados o datos sintéticos diseñados específicamente para el dominio. Cada par de registros evaluados se asigna a una de cuatro categorías:

- Verdaderos positivos (TP): pares clasificados como coincidencia y que efectivamente coinciden.
- Falsos positivos (FP): pares clasificados como coincidencia pero que no coinciden.
- Verdaderos negativos (TN): pares clasificados como no coincidencia y que no coinciden.
- Falsos negativos (FN): pares clasificados como no coincidencia pero que sí coinciden.

A partir de esta clasificación se calculan métricas que permiten evaluar el rendimiento del sistema. En tareas de emparejamiento, donde la mayoría de los pares posibles no son coincidencias, las métricas que dependen de los verdaderos negativos pueden resultar engañosas. Por este motivo se priorizan métricas que se centran en los positivos. En el libro *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*[12], el autor Peter Christen define las siguientes métricas recomendadas y métricas no recomendadas:

Métricas recomendadas

- **Precisión**: Indica qué proporción de las coincidencias detectadas por el sistema son efectivamente correctas. Es útil para medir la confiabilidad del sistema al declarar que dos registros coinciden. Una precisión alta implica que los falsos positivos son pocos.

$$\text{Precisión} = \frac{TP}{TP + FP}$$

- **Recall**:Mide cuántas de las coincidencias reales fueron correctamente identificadas. Es especialmente importante en escenarios donde no detectar una coincidencia implica pérdida de información relevante. Un recall alto indica que el sistema tiene buena cobertura.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-score**:Combina precisión y recall en una sola métrica, destacando los sistemas que logran un equilibrio entre ambos. Es útil cuando se quiere evitar tanto los errores por omisión como por exceso. Estas métricas no dependen del número de verdaderos negativos, lo que las hace adecuadas para contextos con clases desbalanceadas como el emparejamiento de datos, donde las coincidencias reales son una fracción pequeña del total.

$$F_1 = 2 \times \frac{Precisión \times Recall}{Precisión + Recall}$$

Métricas no recomendadas

Métricas como: accuracy, especificidad y tasa de falsos positivos, no son adecuadas en este contexto porque incluyen el número de verdaderos negativos en su cálculo. En emparejamiento de datos, los verdaderos negativos tienden a ser muy numerosos, lo que puede generar valores artificialmente altos incluso cuando el sistema comete muchos errores en la detección de coincidencias. Por esta razón, no aportan información útil sobre el rendimiento real y no se recomiendan como criterio principal de evaluación.

El mismo autor, Peter Christen[12], señala que, debido al fuerte desbalance entre clases en este tipo de tareas, métricas como la accuracy, la especificidad y la tasa de falsos positivos son ineficaces para reflejar el rendimiento real del sistema. Aunque menciona otras opciones como la curva ROC y el área bajo la curva (AUC), advierte que su uso puede resultar engañoso en contextos de emparejamiento, ya que la gran cantidad de verdaderos negativos tiende a producir curvas artificialmente optimistas. El coeficiente de correlación de Matthews (MCC), frecuentemente empleado en clasificación binaria con clases desbalanceadas, no es abordado en este capítulo. En cambio, el enfoque del autor se centra en métricas más robustas e interpretables para data matching, como la precisión, el recall y el F1-score.

Métricas de evaluación para modelos de lenguaje

Las métricas de evaluación para modelos de lenguaje [29] se usan para medir si un modelo ejecuta correctamente una tarea. Algunas requieren una referencia, otras no. Algunas operan a nivel de caracteres, palabras o *embeddings*. Un *embedding* es una representación numérica densa y de dimensión fija que captura características semánticas y sintácticas de una unidad de texto (como una palabra, frase o documento), permitiendo que modelos de aprendizaje automático procesen lenguaje natural de forma eficiente [29].

Hay métricas generales y otras específicas según el tipo de salida. Algunas utilizan un segundo modelo para emitir una evaluación.

Una de las distinciones que se aplica a estas métricas es entre métricas tradicionales y no tradicionales. Las no tradicionales usan representaciones vectoriales o un modelo adicional. Las tradicionales comparan forma y orden. Evalúan coincidencias exactas. La coincidencia exacta y la distancia de edición también forman parte de este grupo.

Métricas no tradicionales

Las métricas no tradicionales utilizan las capacidades avanzadas de los modelos de lenguaje para evaluar la calidad del texto generado más allá de la coincidencia superficial. No se limitan a comparar tokens exactos. Incorporan conocimiento semántico, *embeddings* contextualizados o modelos generativos para emitir juicios de calidad más completos. Estas métricas permiten evaluar propiedades como coherencia, fidelidad o adecuación contextual, incluso cuando el texto generado difiere léxicamente de la referencia. Las métricas más comunes, según Kamath, Keenan, Somers y Sorenson en *Large Language Model: A Deep Dive into Bridging Theory and Practice*[29], son:

- **BERTScore**: compara el texto generado con la referencia utilizando representaciones vectoriales extraídas de modelos preentrenados. Calcula la similitud semántica entre tokens mediante *embeddings* contextualizados. No requiere coincidencia exacta y permite detectar equivalencia de significado aunque se usen palabras distintas.
- **MoverScore**: mejora la sensibilidad frente a paráfrasis y reordenamientos. Utiliza *embeddings* y calcula la distancia mínima necesaria para transformar la distribución semántica del texto generado en la de la referencia. Captura relaciones semánticas globales, no solo similitudes locales.
- **G-Eval**: delega la evaluación en un modelo generativo. Se formula un prompt con instrucciones específicas y el modelo responde con un juicio estructurado sobre propiedades como coherencia, fidelidad o completitud. Esta métrica no depende de una referencia explícita y es adecuada para tareas abiertas.
- **Pass@k**: se utiliza para evaluar tareas con soluciones múltiples válidas, como generación de código. Mide la probabilidad de que al menos una de las k salidas generadas por el modelo sea correcta. Refleja la utilidad práctica del modelo en contextos donde se permiten varios intentos.

Métricas tradicionales

Las métricas tradicionales evalúan la salida generada por el modelo en función de su precisión léxica y sintáctica. Comparan el texto generado con una referencia a partir de coincidencias exactas de palabras, frases o estructuras. Estas métricas han sido ampliamente utilizadas en tareas como traducción automática o resumen, donde el orden y la forma de la salida son relevantes. Su principal limitación es que no capturan significado cuando la redacción varía, incluso si el contenido es correcto. Estas son:

- **BLEU**: calcula la precisión de n-gramas entre la salida y la referencia. Penaliza las salidas más cortas mediante un factor de longitud. Es útil cuando se espera que el modelo reproduzca fragmentos similares en forma y orden [29].

- **ROUGE**: mide el recall de n-gramas, secuencias o estructuras. Evalúa si la salida cubre los elementos clave de la referencia. Se aplica en tareas donde la cobertura del contenido es prioritaria [29].

Otras métricas utilizadas en este grupo, como la coincidencia exacta y la distancia de edición, se vinculan con la detección de errores y serán abordadas en la siguiente sección.

Métricas de similitud estructural

Las métricas de similitud estructural [12] permiten evaluar la calidad de las transformaciones realizadas sobre datos textuales, numéricos o espaciales, sin necesidad de utilizar etiquetas de clase ni tareas de predicción. Se basan en funciones que comparan elementos individuales o pares de registros, y devuelven un valor numérico normalizado entre 0 y 1 que indica el grado de coincidencia.

Son apropiadas para tareas de validación estructural, como la corrección de nombres, la estandarización de direcciones o la reconstrucción de fechas a partir de entradas ruidosas.

Comparadores de texto

Los comparadores de cadenas permiten cuantificar la similitud entre valores textuales. Existen varios enfoques, agrupables según la técnica empleada:

1. **Coincidencia exacta y truncada**. Las funciones de comparación exacta verifican si dos valores textuales coinciden literalmente. Devuelven 1 si son iguales y 0 si difieren. Hay tres variantes:
   - **Exacta**: compara la cadena completa.
   - **Truncada por el inicio**: evalúa coincidencia en los primeros caracteres, útil para prefijos estructurados.
   - **Truncada por el final**: evalúa los últimos caracteres, aplicable cuando hay sufijos comunes o esperados
2. **Distancia de edición**. Las funciones de distancia de edición miden cuántas modificaciones se requieren para transformar una cadena en otra. Se usan para detectar errores, variaciones o deformaciones en valores textuales ya que se basan en una noción de similitud estructural aproximada, donde no se espera coincidencia exacta entre los caracteres. El resultado se normaliza entre 0 y 1, donde 1 indica coincidencia exacta.

   Se distinguen tres funciones principales:
   - **Levenshtein**: cuenta el número mínimo de operaciones necesarias: inserciones, eliminaciones y sustituciones. Es útil para detectar errores ortográficos simples o cambios accidentales de un carácter.

- **Damerau-Levenshtein**: agrega la posibilidad de transposición entre caracteres adyacentes. Se adapta mejor a errores comunes de escritura como invertir dos letras consecutivas.
- **Smith-Waterman**: realiza un alineamiento local entre dos cadenas. En lugar de comparar todo el contenido, busca coincidencias parciales con penalización por errores o vacíos. Es útil cuando el contenido compartido entre cadenas no está alineado al inicio ni al final.

3. **Q-gramas y n-gramas**. Los q-gramas y n-gramas son fragmentos consecutivos de longitud fija que se extraen de una cadena de texto. En el contexto de similitud estructural, se utilizan para comparar la presencia y frecuencia de bloques comunes entre dos valores. Según Peter Christen en *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, se distinguen tres funciones principales, resumidas en la Tabla 2.4.

Función	**Descripción**	**Principio de comparación**	**Uso**
Overlap	Mide el número de q-gramas idénticos compartidos entre dos cadenas	Cuenta cuántos q-gramas son iguales en ambas cadenas	Textos similares con errores menores o diferencias mínimas
Jaccard	Evalúa la proporción de q-gramas comunes frente al conjunto total único	Compara bloques comunes frente al total de bloques únicos	Cadenas con elementos extra o estructuras parcialmente iguales
Dice	Variante de Jaccard que enfatiza la coincidencia relativa	Intersección ponderada respecto a la suma de bloques	Textos breves con fragmentos similares pero no exactos

Tabla 2.4: Funciones de similitud basadas en q-gramas

Comparadores numéricos, fechas y geográficos

Estas funciones evalúan la similitud entre valores continuos o espaciales. A diferencia de los textos, su comparación se basa en la magnitud de la diferencia. Según Peter Christen en *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, se distinguen tres tipos, definidos en la Tabla 2.5.

Tipo de comparación	**Descripción técnica**	**Principio de comparación**	**Aplicación reconocida**
Similitud absoluta	Evalúa si la diferencia entre dos valores es menor que un umbral constante	Comparación directa entre valores reales	Atributos numéricos con error de redondeo
Similitud relativa	Calcula la diferencia proporcional respecto al valor de referencia	Normalización de la diferencia respecto al dato	Datos continuos con escalas dinámicas
Similitud geográfica	Compara ubicaciones por su distancia en el espacio	Transformación de distancia en valor binario	Coordenadas geográficas y ubicaciones aproximadas

Tabla 2.5: Funciones de similitud para valores continuos y espaciales

2.7. Seguridad y privacidad

La protección de datos en sistemas que integran modelos de lenguaje de gran tamaño forma parte del desarrollo responsable. Estos sistemas procesan grandes volúmenes de datos en múltiples fases, lo que introduce riesgos de privacidad que deben ser gestionados mediante estructuras normativas y técnicas específicas. Estos riesgos han sido documentados por el European Data Protection Board en un informe centrado en el uso de LLMs, donde se describen su origen, formas de manifestación, evaluación y medidas de mitigación correspondientes [21].

Fases del ciclo de vida y su impacto en la privacidad

El análisis de riesgos en sistemas de IA debe considerar todo su ciclo de vida, conforme a las normas ISO/IEC 22989 e ISO/IEC 5338. Este ciclo se compone de las siguientes fases, cada una con riesgos de privacidad asociados:

- **Diseño**: se definen las fuentes de datos y estrategias de tratamiento. La inclusión de datos sensibles sin mecanismos de anonimización representa una vulnerabilidad temprana.
- **Preparación de datos**: se recopila, limpia y transforma la información. Aquí pueden introducirse datos personales identificables, errores de anonimización o sesgos estructurales que afecten la equidad del modelo.
- **Entrenamiento**: el modelo aprende patrones a partir de los datos preparados. En esta etapa puede memorizar fragmentos sensibles, lo que supone un riesgo de reproducción involuntaria.

- **Validación**: se evalúa el comportamiento del modelo con conjuntos de prueba, que pueden contener datos reales no anonimizados.
- **Despliegue**: el sistema interactúa con usuarios y datos en tiempo real. Los prompts y las respuestas generadas pueden contener o derivar información personal no prevista.
- **Operación y mantenimiento**: se registran logs, interacciones y métricas. Si no se anonimizan, estos registros pueden contener trazas identificables.
- **Actualización y retirada**: se modifican, reentrenan o eliminan modelos. El tratamiento incorrecto de datos históricos o residuales puede mantener riesgos persistentes.

Esta estructura permite identificar puntos críticos donde deben aplicarse medidas técnicas y organizativas específicas para garantizar la seguridad del tratamiento.

Categorías de riesgos de privacidad en LLMs

Los riesgos de privacidad en sistemas con LLMs, según el European Data Protection Board (EDPB)[21], pueden clasificarse en función del momento en que se materializan y de la forma en que se manifiestan:

- **Riesgos en el entrenamiento**: inclusión de datos personales en los conjuntos de entrenamiento sin validación previa, o modelado sobre datos no anonimizados.
- **Riesgos en la inferencia**: generación de salidas que revelan información sensible, ya sea por memorias del modelo o por mal diseño de prompts.
- **Riesgos en el feedback loop**: almacenamiento de interacciones sin anonimización, uso de respuestas del usuario como datos para reajuste del modelo.
- **Riesgos en RAG**: por sus siglas en inglés Retrieval-Augmented Generation, uso de bases de conocimiento con datos identificables sin control de acceso ni trazabilidad.
- **Riesgos en logs y monitoreo**: persistencia de datos sensibles en registros de actividad si no se purgan ni cifran adecuadamente.

Esta clasificación permite delimitar el alcance de cada riesgo y definir medidas proporcionales a su nivel de criticidad.

Evaluación del riesgo de privacidad

La evaluación de riesgos en estos sistemas se basa en dos variables fundamentales:

- **Probabilidad de ocurrencia**: se refiere a la posibilidad de que el riesgo se materialice, considerando factores como el volumen de datos tratados, la naturaleza de los datos, ya sean sensibles o no, y la presencia o ausencia de medidas de mitigación.

- **Severidad del impacto**: mide el daño potencial sobre los derechos y libertades de los afectados, considerando el tipo de dato, el contexto de uso y la finalidad del tratamiento.

Combinando ambas dimensiones, se puede clasificar cada riesgo en un nivel bajo, medio o alto, lo que permite priorizar acciones de control. Esta evaluación no sustituye a una evaluación de impacto relativa a la protección de datos según el artículo 35 del RGPD[1]., pero la complementa al centrarse en los riesgos específicos de los sistemas basados en modelos de los LLMs.

Medidas técnicas y organizativas de mitigación

El tratamiento de los riesgos identificados requiere la aplicación de medidas concretas. Estas medidas, según el EDPB[21] pueden ser clasificadas en:

- **Medidas técnicas**:
  - Cifrado de datos en tránsito y en reposo.
  - Segmentación de red y aislamiento del sistema de IA.
  - Eliminación segura de datos y políticas de retención.
  - Control de acceso por roles y registro de actividad.
  - Auditoría automatizada de logs e inferencias.
- **Medidas organizativas**:
  - Políticas de minimización de datos.
  - Registro de finalidades y criterios de licitud del tratamiento.
  - Prohibición de reutilización de datos sin consentimiento.
  - Evaluación y revisión continua de los riesgos residuales.

Estas acciones deben integrarse desde la fase de diseño, conforme al principio de protección de datos desde el diseño y por defecto según el artículo 25 del RGPD[2], y alinearse con la obligación de garantizar la seguridad del tratamiento establecida en el artículo 32 del mismo reglamento[3].

[1]El artículo 35 del Reglamento General de Protección de Datos regula la obligación de realizar una evaluación de impacto cuando un tratamiento pueda suponer un alto riesgo para los derechos y libertades de las personas, especialmente en casos de tecnologías nuevas, decisiones automatizadas o tratamiento a gran escala [42, pp. 30]

[2]El artículo 25 del Reglamento General de Protección de Datos exige que se apliquen medidas técnicas y organizativas apropiadas desde el diseño y por configuración predeterminada, de forma que solo se traten los datos personales necesarios para cada finalidad [42, pp. 23].

[3]El artículo 32 del Reglamento General de Protección de Datos establece la obligación de aplicar medidas adecuadas para garantizar la seguridad del tratamiento, considerando el riesgo y medidas como el cifrado, control de accesos, resiliencia y evaluación periódica [42, pp. 28].

Supervisión y revisión continua

El ciclo de gestión de riesgos en sistemas basados en LLMs requiere mecanismos de revisión periódica y supervisión continua. Esto incluye:

- Registro actualizado de riesgos y controles.
- Seguimiento del comportamiento del modelo ante nuevos datos.
- Evaluación de impacto ante cambios en la configuración o el flujo de datos.
- Validación de la eficacia de las medidas aplicadas frente a vulnerabilidades emergentes.

La revisión constante permite mantener el riesgo residual dentro de umbrales aceptables, evitando su acumulación y reduciendo la posibilidad de incidentes derivados del envejecimiento del sistema o del cambio de contexto.

Capítulo 3

Estado del arte

3.1. Búsqueda en literatura académica

La revisión sistemática de literatura (SLR, por sus siglas en inglés) es un tipo de estudio secundario que se basa en un proceso estructurado y replicable para identificar, analizar y sintetizar la evidencia disponible sobre una pregunta de investigación concreta. Fue introducida en ingeniería de software como parte del enfoque de ingeniería basada en evidencia, el cual busca integrar los mejores resultados de investigación con la experiencia práctica y los valores del contexto profesional para mejorar la toma de decisiones técnicas [32].

Este tipo de revisión busca combinar resultados de estudios previos para responder a una pregunta concreta. Para ello, aplica filtros sistemáticos, valora la calidad de la información y sintetiza los datos de forma objetiva. Permite obtener respuestas más precisas y firmes que las revisiones sin estructura metodológica. La revisión sistemática se apoya en la evidencia, tanto cualitativa como cuantitativa, y su utilidad depende de la claridad en cada fase: planteamiento de la pregunta, selección de fuentes, evaluación de artículos y análisis de resultados [44].

Con el objetivo de responder a la pregunta de investigación RQ1, que plantea qué enfoques recientes han explorado el uso de modelos generativos en tareas de limpieza y mejora de calidad de datos estructurados, se usó un enfoque de revisión sistemático en tres bases de datos académicas reconocidas: Scopus, IEEE Xplore y Web of Science.

La búsqueda incial se enfocó en publicaciones recientes, limitando los resultados a los años 2023, 2024 y 2025, con el fin de identificar los trabajos más actuales sobre la aplicación de inteligencia artificial generativa. Esto considerando que el uso de IA generativa recién comenzó a popularizarse en publicaciones académicas a partir de 2023, tras el lanzamiento público de ChatGPT en noviembre de 2022 [54].

En la primera exploración se usaron los siguientes filtros:

- Años: 2023, 2024 y 2025

- Términos utilizados:
  - data
  - quality OR cleaning OR cleansing OR profiling OR preparation
  - LLM OR AI OR GenAI OR "artificial intelligence" OR "generative artificial intelligence"

Estrategia de búsqueda

El protocolo de la revisión sistemática define las bases metodológicas para garantizar la transparencia, reproducibilidad y coherencia del proceso. Incluye la selección de fuentes, las estrategias de búsqueda, los criterios de inclusión y exclusión, y la herramienta de extracción y análisis de datos[9]. Su función principal es guiar la ejecución de la revisión de forma controlada y sin introducir sesgos por decisiones no documentadas.

Criterios de inclusión definidos:

- Idioma: estudios en inglés o español.
- Tipo de documento: artículos científicos revisados por pares y actas de congresos.
- Periodo: publicaciones entre 2022 y 2025.
- Disponibilidad: acceso completo al texto del estudio.
- Contenido: estudios que incluyan tareas de limpieza o preparación de datos estructurados mediante modelos generativos o modelos de lenguaje.

Criterios de exclusión definidos:

- Estudios duplicados entre múltiples fuentes.
- Documentos sin acceso al texto completo.
- Literatura gris, tesis, libros o informes no revisados por pares.
- Publicaciones fuera del rango temporal definido.
- Estudios que no incluyan ninguna mención a técnicas de procesamiento automático con modelos generativos.

La búsqueda se planificó para ejecutarse en las bases de datos Scopus, IEEE Xplore y Web of Science, seleccionadas por su cobertura en informática, ingeniería y tecnologías de la información. Las cadenas de búsqueda se construyeron mediante operadores booleanos, combinando términos relacionados con limpieza de datos, modelos de lenguaje y preparación automatizada.

3.2. Resultados de la SLR

Resultados iniciales

Los querys para cada base de datos y sus resultados fueron:

Base de datos	**Query aplicado**	**Años**	**Resultados**
Scopus	TITLE-ABS-KEY (data AND (quality OR cleaning OR cleansing OR profiling OR preparation) AND (llm OR ai OR genai OR "artificial intelligence" OR "generative artificial intelligence")) AND (PUBYEAR = 2023 OR PUBYEAR = 2024 OR PUBYEAR = 2025)	2023-2025	20165
IEEE Xplore	((("All Metadata":data) AND (("All Metadata":quality) OR ("All Metadata":cleaning) OR ("All Metadata":cleansing) OR ("All Metadata":profiling) OR ("All Metadata":preparation)) AND (("All Metadata":LLM) OR ("All Metadata":AI) OR ("All Metadata":GenAI) OR ("All Metadata":"artificial intelligence") OR ("All Metadata":"generative artificial intelligence"))))	2023-2025	17038
Web of Science	TS=(data AND (quality OR cleaning OR cleansing OR profiling OR preparation) AND (LLM OR AI OR GenAI OR "artificial intelligence" OR "generative artificial intelligence"))	2023-2025	52283

Tabla 3.1: Resultados iniciales por base de datos

Se observó que los resultados obtenidos son muy generales y numerosos, lo que dificulta su exploración. Muchos documentos están relacionados con áreas como medicina, biología, energía, entre otras, sin un enfoque claro en calidad de datos. Por lo tanto, se procede a refinar más las búsquedas.

Resultados usando refinamiento avanzado

La tabla 3.2 resume los resultados obtenidos en cada base de datos tras aplicar los filtros avanzados definidos en la estrategia de búsqueda.

Base de datos	**Query aplicado**	**Años**	**Resultados**
Scopus	((TITLE-ABS-KEY (cleaning AND data AND using AND large AND language AND models) AND KEY (generative OR artificial OR intelligence OR data OR cleaning OR data OR preparation))) AND PUBYEAR > 2022 AND PUBYEAR < 2026 AND (LIMIT-TO (EXACTKEYWORD , “Metadata”) OR LIMIT-TO (EXACTKEYWORD , “Data Preparation”) OR LIMIT-TO (EXACTKEYWORD , “Data Cleaning”) OR LIMIT-TO (EXACTKEYWORD , “Data Accuracy”))	2022-2025	36
IEEE Xplore	((((“All Metadata”:“cleaning” AND “All Metadata”:“data” AND “All Metadata”:“using” AND “All Metadata”:“large” AND “All Metadata”:“language” AND “All Metadata”:“models”) AND ((“All Metadata”:“generative”) OR (“All Metadata”:“artificial”) OR (“All Metadata”:“intelligence”) OR (“All Metadata”:“data”) OR (“All Metadata”:“cleaning”) OR (“All Metadata”:“preparation”))) AND ((“All Metadata”:“metadata”) OR (“All Metadata”:“data preparation”) OR (“All Metadata”:“data cleaning”) OR (“All Metadata”:“data accuracy”)))	2022-2025	15
Web of Science	TS=((cleaning AND data AND using AND large AND language AND models) AND (generative OR artificial OR intelligence OR data OR cleaning OR data OR preparation)) AND TS=(“metadata” OR “data preparation” OR “data cleaning” OR “data accuracy”)	2022-2025	31

Tabla 3.2: Resultados refinados por base de datos

El resultado fue de 36 estudios en Scopus, 15 en IEEE Xplore y 31 en Web of Science, con un total de 82 publicaciones.

Tras aplicar los criterios de eliminación, se obtuvo lo siguiente:

- Duplicados: se eliminaron 17 estudios repetidos, tomando como base principal Scopus.
- Idioma: solo se consideraron estudios en inglés o español. Se eliminó 1 estudio.

- Disponibilidad: se eliminaron 13 estudios por no tener acceso al texto completo.
- Tipo de documento: solo se incluyeron artículos y conferencias. Se eliminó 1 tesis.

El resultado final fue de 50 estudios.

Estudios primarios seleccionados

Se aplicaron criterios temáticos, y tras una revisión exploratoria de los textos completos, se seleccionaron 16 estudios primarios que pueden responder a las preguntas de investigación. La tabla 3.3 presenta su categorización según tipo de contribución; considerando si se trata de una herramienta; una propuesta metodológica; un caso de aplicación generalizable,que parte de un contexto concreto pero puede adaptarse a otros similares; o un caso generalizado, validado en múltiples dominios.

Código	Título del estudio	Año	Tipo	Categoría
[BLD+23]	Ask Language Model to Clean Your Noisy Translation Data	2023	Conferencia	Caso que intenta generalizar
[NAE+24]	RetClean: Retrieval-Based Data Cleaning Using LLMs and Data Lakes	2024	Conferencia	Herramienta
[MAD+24]	Cleaning Semi-Structured Errors in Open Data Using Large Language Models	2024	Conferencia	Caso generalizado
[HSL+24]	Application of Large Language Models in Chemistry Reaction Data Extraction and Cleaning	2024	Conferencia	Caso que intenta generalizar
[GGBR24]	Harnessing GPT for Data Transformation Tasks	2024	Conferencia	Metodología
[PSA+24]	Improving Understandability and Control in Data Preparation: A Human-Centered Approach	2024	Conferencia	Metodología
[MR24]	Leveraging Structured and Unstructured Data for Tabular Data Cleaning	2024	Conferencia	Caso generalizado
[ZL24]	Smartphone Usage Data Cleaning Using LLM-Based Processing	2024	Conferencia	Caso que intenta generalizar
[BADG25]	LLMClean: Context-Aware Tabular Data Cleaning via LLM-Generated OFDs	2025	Conferencia	Herramienta
[NdTB25]	Improving Data Cleaning by Learning From Unstructured Textual Data	2025	Artículo	Metodología
[BDVI24]	Improving the Quality of Diabetic Data with LLM-driven Cleaning Techniques	2024	Conferencia	Caso que intenta generalizar
[NZM+24]	IterClean: An Iterative Data Cleaning Framework with Large Language Models	2024	Conferencia	Herramienta & Metodología
[ZNS+24]	Self-Repairing Data Scraping for Websites	2024	Conferencia	Herramienta
[CTFL23]	Demystifying Artificial Intelligence for Data Preparation	2023	Conferencia	Metodología
[WLW23]	Sudowoodo: Contrastive Self-supervised Learning for Multi-purpose Data Integration and Preparation	2023	Conferencia	Herramienta
[CLW+24]	PreparedLLM: effective pre-pretraining framework for domain-specific large language models	2024	Artículo	Metodología

Tabla 3.3: Estudios primarios seleccionados y su categorización

Del total, 5 estudios fueron clasificados como herramientas, 5 como propuestas metodológicas, 4 como casos que intentan generalizar y 2 como casos generalizados.

Literatura Gris

Se incluyeron artículos de arXiv al no estar presentes en bases como Scopus o IEEE. Al no contar con revisión por pares ni DOI, se consideran literatura gris. Aun así, se valoraron por su actualidad y por aportar enfoques significativos al tema de mejora de calidad de datos con modelos de lenguaje.

Código	**Título del estudio**	**Año**	**Categoría**
[BvdLN+23]	OpusCleaner and OpusTrainer, open source toolkits for training Machine Translation and Large language models	2023	Herramienta
[EY24]	Organic Data-Driven Approach for Turkish Grammatical Error Correction and LLMs	2024	Metodología
[ZHW24]	Data Cleaning Using Large Language Models	2024	Herramienta / Metodología
[LFT24]	AutoDCWorkflow: LLM-based Data Cleaning Workflow Auto-Generation and Benchmark	2024	Herramienta / Metodología
[LLT25]	Too Noisy To Learn: Enhancing Data Quality for Code Review Comment Generation	2025	Caso generalizado
[SLW+25]	DCAD-2000: A Multilingual Dataset across 2000+ Languages with Data Cleaning as Anomaly Detection	2025	Metodología
[MPSD25]	Are Large Language Models Good Data Preprocessors?	2025	Metodología
[LQL+25]	MathClean: A Benchmark for Synthetic Mathematical Data Cleaning	2025	Metodología

Tabla 3.4: Estudios relevantes provenientes de literatura gris (arXiv) y su categorización

Se identificaron 8 estudios pre-publicados según la temática en arXiv. De ellos, 4 fueron clasificados como Metodología, 2 como Herramienta & Metodología, 2 como Caso que intenta generalizar y 1 como Herramienta.

No se aplicó la técnica complementaria de *snowballing*, que consiste en revisar manualmente las referencias citadas en los estudios incluidos para identificar publicaciones adicionales que no aparecieron en las búsquedas iniciales [9]. Esta técnica puede ser útil para ampliar el conjunto de estudios, pero requiere una evaluación posterior para determinar su relevancia y evitar duplicidades.

Análisis de estudios primarios

La Tabla 3.5 muestra que la categoría Metodología es la más frecuente, con un 37,5 % del total de estudios. Según el gráfico de burbujas en la Figura 3.1, el 16,7 % de estos trabajos se concentra en 2024 y otro 16,7 % en 2025, siendo los años con mayor representación en esta categoría.

Categoría	Cantidad	Porcentaje
Metodología	9	37,5 %
Caso que intenta generalizar	5	20,8 %
Herramienta	4	16,7 %
Caso generalizado	3	12,5 %
Herramienta & Metodología	3	12,5 %

Tabla 3.5: Distribución de estudios primarios por categoría

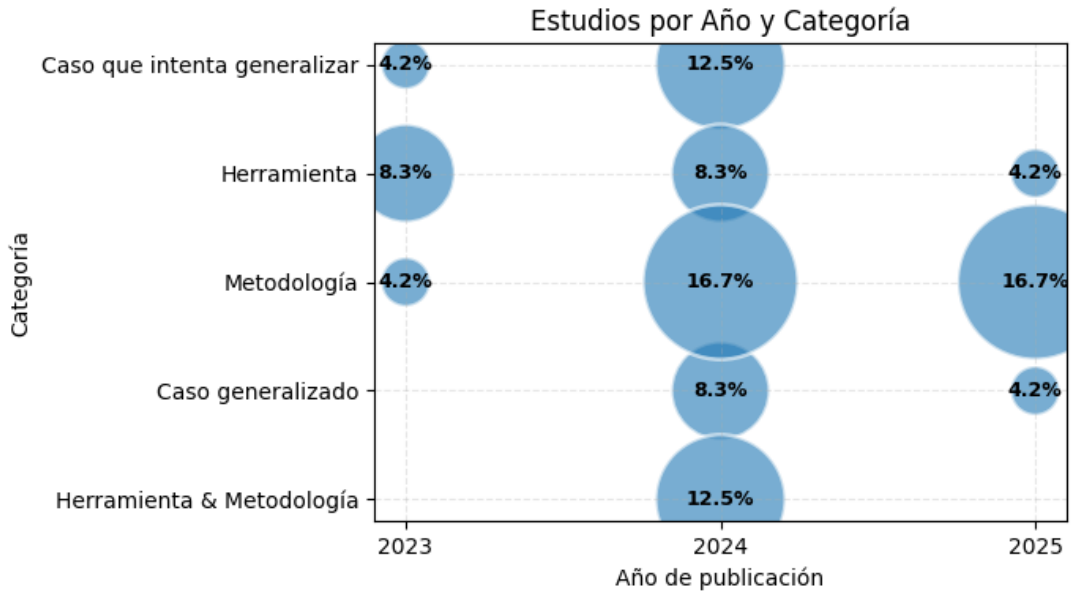

Figura 3.1: Distribución de los estudios primarios en año de publicación y categoría

A continuación, se presentan los resúmenes de los estudios analizados:

Metodología

Ask Language Model to Clean Your Noisy Translation Data [BLD+23]:
Propone una metodología asistida por LLMs para limpiar valores erróneos, inconsistentes o atípicos en datos estructurados. Combina contexto semántico, reglas

estadísticas y conocimiento externo para generar sugerencias de corrección. Evalúa su eficacia en distintos dominios.

Harnessing GPT for Data Transformation Tasks [GGBR24]:
Presenta una metodología basada en GPT-4 para tareas de transformación de datos estructurados. Propone una plantilla de prompts reutilizable y evalúa su desempeño en tareas como limpieza, extracción y reformateo. La metodología se implementa en una biblioteca Python (DTLM) y se compara contra herramientas clásicas en un benchmark especializado.

Improving Understandability and Control in Data Preparation [PSA+24]:
Propone una metodología centrada en el usuario para diseñar entornos de preparación de datos explicables y personalizables. Define requisitos mediante entrevistas, escenarios y sesiones think-aloud, y extiende una plataforma con explicaciones generadas por LLMs, visualizaciones interactivas y control granular del proceso. La metodología busca mejorar la comprensión y el control del usuario sobre las acciones de calidad de datos.

Improving Data Cleaning by Learning From Unstructured Textual Data [NdTB25]:
Propone una metodología basada en aprendizaje multitarea con LLMs para limpiar datos estructurados utilizando descripciones textuales no estructuradas. El modelo aprende patrones desde texto libre y mejora tanto la imputación como la detección de errores.

Demystifying Artificial Intelligence for Data Preparation [CTFL23]:
Propone una metodología basada en una taxonomía funcional para integrar PLMs y modelos generativos en tareas de preparación de datos. Aborda limpieza, vinculación y transformación con modelos como GPT-3 o Retro, destacando aspectos como explicabilidad y control humano.

Effective Pre-Pretraining Framework for Domain-Specific LLMs [CLW+24]:
Propone una metodología de pre-preentrenamiento que automatiza la limpieza, selección y estructuración de datos para LLMs específicos de dominio. Mejora la calidad del corpus antes del fine-tuning mediante recetas de datos, expansión de vocabulario y técnicas de inicialización de embeddings.

Are Large Language Models Good Data Preprocessors? [MPSD25]:
Propone una metodología experimental para evaluar LLMs como preprocesadores de texto. Compara GPT-4, LLaMA 3.1 y Sonnet 3.5 en la limpieza de captions generados por modelos multimodales. Analiza su impacto en una tarea compleja de clasificación de memes persuasivos, incorporando pruebas estadísticas para valorar su efectividad.

MathClean: A Benchmark for Synthetic Mathematical Data Cleaning [LQL+25]:
Propone una metodología basada en aprendizaje multitarea con LLMs para limpiar

datos estructurados utilizando texto libre como fuente adicional. Combina entradas estructuradas y no estructuradas para mejorar la imputación y detección de errores. Evalúa su eficacia en tareas de limpieza sobre datasets reales.

Caso que intenta generalizar

Application of Large Language Models in Chemistry Reaction Data Extraction and Cleaning [HSL+24]:
Estudia el uso de LLMs para extraer y limpiar datos de reacciones químicas no estructurados. Evalúa GPT-4 y LLaMA-2 con fine-tuning y prompt tuning. Propone una metodología específica para este dominio, con un verificador factual que garantiza la precisión de la información extraída.

Smartphone Usage Data Cleaning Using LLM-Based Processing [ZL24]:
Utiliza LLMs para limpiar y transformar datos recolectados por smartphones. Automatiza la inferencia de etiquetas, agrupación semántica y eliminación de ruido. Centrado en datos sensibles de comportamiento humano.

Improving the Quality of Diabetic Data with LLM-driven Cleaning Techniques [BDVI24]:
Aplica LLMs a la mejora de datasets médicos sobre diabetes. Automatiza la detección de valores incoherentes y outliers. Evalúa el impacto en análisis de predicción médica posterior.

Too Noisy To Learn: Enhancing Data Quality for Code Review Comment Generation [LLT25]:
Analiza cómo el ruido en los datos de revisión de código afecta la generación automática de comentarios. Propone estrategias de limpieza para reducir duplicados, errores y alineaciones inconsistentes en los pares código -comentario. Evalúa el impacto de estos filtros en modelos generativos y demuestra que la calidad de los datos modifica directamente los resultados.

Caso generalizado

Cleaning Semi-Structured Errors in Open Data Using Large Language Models [MAD+24]:
Aborda la limpieza de errores en datos semi-estructurados usando aprendizaje en contexto con LLMs. Se centra en datos legales en formato XML. Evalúa el rendimiento de diferentes prompts para corrección automática.

Leveraging Structured and Unstructured Data for Tabular Data Cleaning [MR24]:
Usa tanto datos estructurados como no estructurados para limpiar datos tabulares. Propone un framework que mejora imputación y detección de errores usando contexto textual. Aumenta la calidad de la limpieza más allá de reglas fijas.

DCAD-2000: A Multilingual Dataset across 2000+ Languages with Data Cleaning as Anomaly Detection [SLW+25]:
Propone una técnica de limpieza de datos que se basa en detección de anomalías sin umbrales predefinidos. Aplica el enfoque a un corpus de más de 2,000 lenguas. Su escala y metodología innovadora muestran cómo los LLMs pueden aplicarse en la limpieza multilingüe a gran escala.

Herramienta

RetClean: Retrieval-Based Data Cleaning Using LLMs and Data Lakes [NAE+24]:
Presenta un sistema de limpieza de datos que combina RAG con LLMs, ideal para datos empresariales privados. Muestra cómo aprovechar data lakes para mejorar sugerencias de limpieza. Soporta tres escenarios según privacidad y tipo de modelo.

LLMClean: Context-Aware Tabular Data Cleaning via LLM-Generated OFDs [BADG25]:
Introduce LLMClean, un sistema que genera dependencias funcionales a partir de contexto textual usando LLMs. Automatiza tareas clásicas como detección de errores y valores faltantes. Evalúa resultados en múltiples datasets.

Self-Repairing Data Scraping for Websites [ZNS+24]:
Desarrolla un sistema para corregir automáticamente errores en scraping web usando LLMs. Detecta cuándo fallan las reglas de extracción y las adapta con contexto. Aumenta robustez frente a cambios estructurales en sitios web.

OpusCleaner and OpusTrainer: open source toolkits for training Machine Translation and Large Language Models [BvdLN+23]:
Presenta dos herramientas open source orientadas a facilitar la descarga, limpieza y preprocesamiento de datos de traducción. Automatizan tareas como detección de ruido, mezcla de corpus y aumento de datos durante el entrenamiento. Aplican filtros y transformaciones para mejorar la robustez de los modelos.

Sudowoodo: Contrastive Self-supervised Learning for Multipurpose Data Integration and Preparation [WLW23]:
Presenta un modelo auto-supervisado que prepara datos para múltiples tareas como integración y limpieza. No depende de reglas fijas, sino de similitud semántica. Entrenado con contraste de ejemplos limpios vs ruidosos.

Herramienta & Metodología

IterClean: An Iterative Data Cleaning Framework with Large Language Models [NZM+24]:
Propuesta de un sistema iterativo donde el modelo mejora su limpieza con cada ciclo de feedback. Usa LLMs para identificar errores y reescribir valores incorrectos. Incluye evaluación humana y automática.

AutoDCWorkflow: LLM-based Data Cleaning Workflow Auto-Generation and Benchmark [LFT24]:
Este trabajo propone un pipeline automatizado donde agentes LLM generan, seleccionan y ejecutan flujos de limpieza de datos. Evalúa distintos enfoques sobre un benchmark propio. Explora el uso de LLMs como planificadores automáticos en tareas de calidad de datos.

Data Cleaning Using Large Language Models [ZHW24]:
Propone Cocoon, un sistema que combina detección estadística con razonamiento semántico usando LLMs. Descompone tareas complejas de limpieza en subtareas más manejables: duplicados, valores atípicos, errores de tipo, entre otros. Ejecuta la limpieza vía SQL y permite interacción humana en el proceso.

Capítulo 4

Caso de estudio

4.1. RetClean como herramienta base

Después de revisar la literatura relacionada, se seleccionó RetClean como sistema base. RetClean [39] es una herramienta desarrollada por investigadores del Qatar Computing Research Institute (QCRI) y la Hong Kong University of Science and Technology (HKUST) para la reparación de datos tabulares. Combina modelos generativos con recuperación de ejemplos desde data lakes. Permite cargar archivos CSV, seleccionar una columna objetivo, definir valores a corregir, usar columnas pivote como contexto y configurar modelos LLM con o sin recuperación aumentada.

Esta herramienta fue presentada en un estudio publicado en diciembre de 2024 en *Proceedings of the VLDB Endowment*[1]. Desde entonces, ha sido citado en investigaciones que aplican RAG para imputación de valores en tablas con datos faltantes [46], y en trabajos sobre construcción de data lakes a escala masiva con soporte para limpieza automática [31].

No fue posible replicar directamente el caso presentado en el estudio original [39], ya que la versión actual del repositorio no incluye el modelo afinado por fine-tuning ni las funciones de filtrado aplicadas durante la recuperación, ambas utilizadas en ese experimento. En su lugar, se empleó el ejemplo funcional disponible en el repositorio oficial, que utiliza un mini dataset de fútbol para pruebas básicas. Este ejemplo permitió validar el funcionamiento general del sistema y comprobar sus capacidades con distintas configuraciones.

La Figura 4.1 muestra la interfaz descrita en el artículo original, donde se observa la presencia de un modelo fine-tuned y filtros configurables para el proceso de recuperación. En contraste, la Figura 4.2 corresponde a la versión actual disponible en GitHub, en la que estas opciones ya no están presentes.

[1] https://dl.acm.org/toc/pvldb/2024/17/12

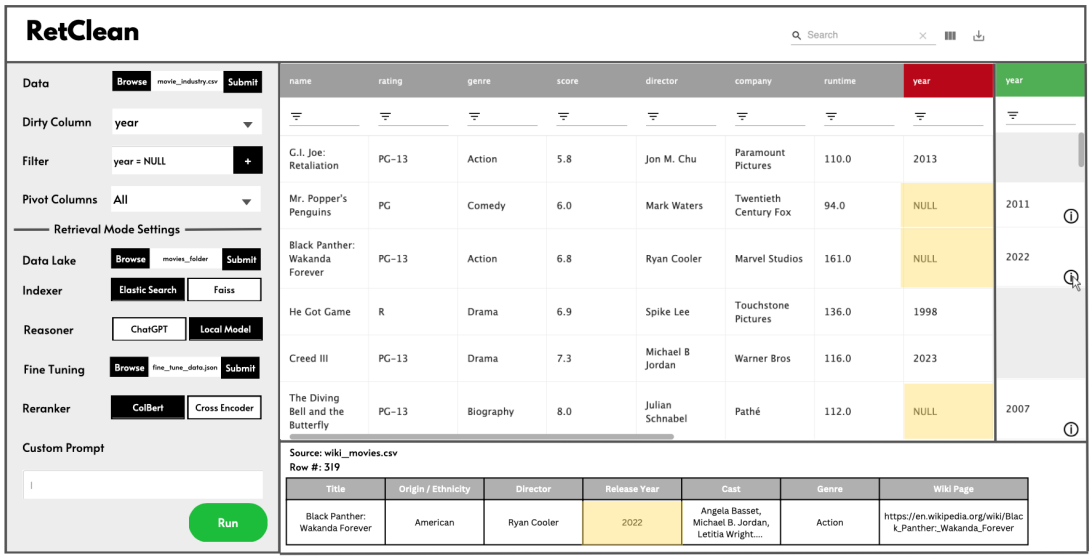

Figura 4.1: Interfaz de RetClean en el estudio original.

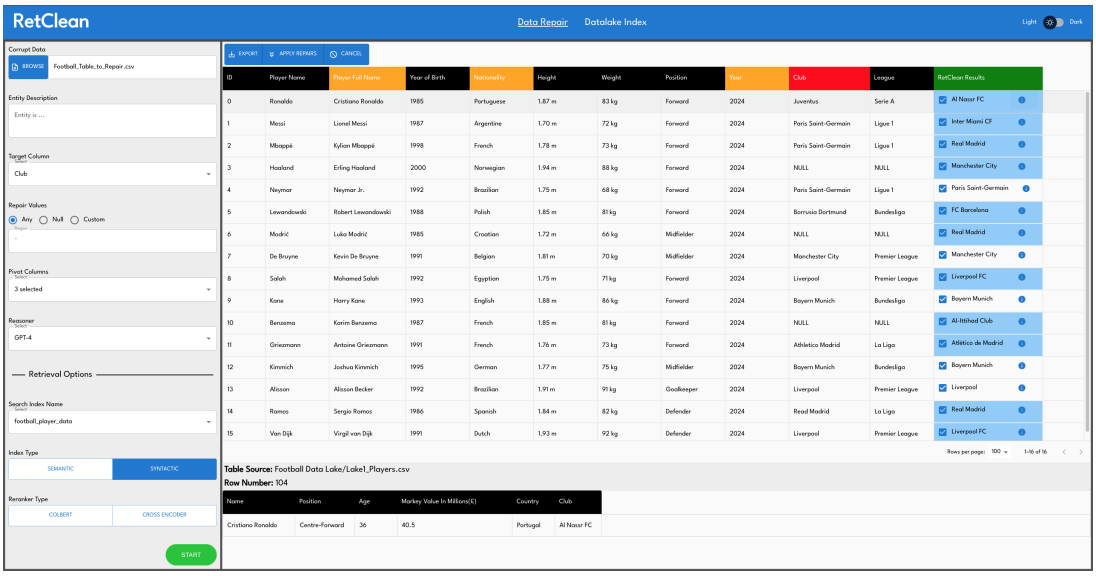

Figura 4.2: Versión actual del repositorio de RetClean.

En este contexto, se encontró que RetClean soporta tres configuraciones principales. La primera utiliza únicamente un modelo generativo sin contexto externo. La segunda añade recuperación de ejemplos desde un índice, combinando el modelo con registros relevantes obtenidos de un data lake. La tercera ejecuta el mismo flujo usando modelos locales, manteniendo los datos en entornos controlados.

Este sistema fue elegido por cuatro motivos:

- Su código es abierto y bajo licencia MIT, disponible en: https://github.com/qcri/RetClean.
- Su diseño modular permite incorporar modelos propios y fuentes de datos sin alterar el flujo general.
- Su alcance cubre desde completar valores vacíos hasta reparar datos usando ejemplos recuperados y con trazabilidad.
- Permite usar modelos locales, lo que hace posible trabajar incluso con datos sensibles, ya que no se derivan a servicios externos.

4.2. Análisis de riesgos

Este análisis se basa en el marco de riesgos descrito en el documento AI Privacy Risks and Mitigations in Large Language Models, respaldado por el European Data Protection Board [21]. Se identificaron los riesgos aplicables a un sistema local como RetClean, que opera sin conexión a internet, sin usuarios externos y sin datos personales reales. Se consideraron aspectos como la inferencia, la interfaz local, el almacenamiento temporal y el entorno de ejecución en contenedores.

En la Tabla 4.2 se muestran los riesgos identificados junto con la medida correspondiente en cada caso. Se distinguen tres situaciones: medidas ya implementadas, medidas no implementadas hasta el momento, y riesgos que no requieren cambios porque ya están cubiertos por el diseño actual del sistema.

ID	Riesgo	Fuente del riesgo	Impacto potencial	Nivel	Medidas aplicadas
R1	Contenido sensible en salidas	Memoria del modelo	El modelo puede generar nombres, correos o identificadores ficticios no presentes en el CSV	Alto	No se ha implementado validación post-inferencia.
R2	Persistencia de archivos temporales	Almacenamiento en disco	Riesgo de reutilización o acceso posterior a los archivos cargados	Medio	Se eliminan automáticamente cada 60 minutos. Script de limpieza creado.
R3	Captura en logs	Registro en contenedor	El contenido procesado podría quedar registrado fuera del control del usuario	Bajo	Ninguna. Solo existen logs en tiempo real que se eliminan al apagar el contenedor.
R4	Volúmenes persistentes en Docker	Configuración del contenedor	Datos podrían quedar almacenados en disco a través de volúmenes no deseados	Medio	Se desactivaron volúmenes comentando líneas en Docker Compose.
R5	Accesos simultáneos no gestionados	Concurrencia en interfaz web	Dos peticiones paralelas podrían interferir entre sí	Bajo	Ninguna. El diseño solo permite procesar una petición a la vez.
R6	Visualización directa de datos cargados	Interfaz gráfica local	Los datos cargados se visualizan sin restricciones desde el navegador	Medio	Ninguna. Uso limitado al entorno local.
R7	Inferencias incoherentes sin contexto	Prompt del modelo	El modelo puede devolver datos inexistentes en el CSV cargado	Medio	No hay validación semántica para verificar coherencia entre salida y CSV.
R8	Acceso del modelo a rutas del sistema	Arquitectura general	Riesgo de fuga o acceso a archivos fuera del entorno previsto	Bajo	Los modelos se ejecutan en contenedores aislados.

Tabla 4.1: Análisis de riesgos de privacidad en RetClean

4.3. Adaptaciones implementadas

Con el fin de adaptar la herramienta original al entorno experimental, se realizaron modificaciones tanto en la configuración del entorno de ejecución como en el código fuente. Todas las adaptaciones están disponibles en el repositorio: https://gitlab.inf.uva.es/micplua/retclean-analisis-tfm.git.

Actualización del archivo de licencia

El archivo `LICENSE` fue modificado para incluir los términos de la licencia Creative Commons Attribution-NonCommercial 4.0 International, además de la licencia MIT original. Esta nueva cláusula define que las modificaciones realizadas pueden ser reutilizadas con fines no comerciales, siempre que se mantenga la atribución correspondiente [13]. El texto completo de la licencia fue incorporado directamente en el archivo y corresponde al contenido oficial disponible[2].

Configuración del entorno y contenedores Docker

Para evitar la descarga repetida de modelos tras cada reinicio, se configuró un volumen persistente en el contenedor `ollama`. El cambio se realizó en el archivo `docker-compose.yml`:

```
ollama:
  build: ./ollama
  volumes:
    - ./ollama:/app
    - ./models:/root/.ollama/
```

Se habilitó soporte para GPU con el objetivo de acelerar el cómputo durante la ejecución de modelos:

```
ollama:
  deploy:
    resources:
      reservations:
        devices:
          - driver: nvidia
            count: 1
            capabilities: [gpu]
```

Para mantener el entorno limpio y facilitar pruebas reproducibles, se deshabilitaron los volúmenes persistentes en los contenedores `elasticsearch` y `qdrant` comentando las siguientes líneas:

```
# volumes:
#   - es_storage:/usr/share/elasticsearch/data

# volumes:
#   - qdrant_storage:/qdrant/storage
```

Se desarrolló el script `entrypoint.sh` para automatizar la descarga de modelos al iniciar el contenedor. Este script se integra mediante el archivo `ollama/dockerfile`:

```
FROM ollama/ollama

COPY ./entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

[2] https://creativecommons.org/licenses/by-nc/4.0/legalcode.txt

Contenido del script `entrypoint.sh`:

```
#!/bin/bash

./bin/ollama serve &
pid=$!
sleep 5

models=("llama3.1:8b" "mistral:7b" "llava:7b")

for model in "${models[@]}"; do
  echo "Pulling model '$model'"
  ollama pull "$model"
done

wait $pid
```

Se añadió la variable de entorno `OLLAMA_KEEP_ALIVE` con valor `10` para controlar la persistencia de los modelos descargados:

```
ollama:
  ...
  environment:
    OLLAMA_KEEP_ALIVE: "10m"
  ...
```

Integración de modelos personalizados y externos

El script `entrypoint.sh` permite descargar automáticamente tres modelos al iniciar el contenedor:

```
models=("llama3.1:8b" "mistral:7b" "llava:7b")
```

Para facilitar la integración de modelos personalizados, se creó el archivo `custom_model.py`, que define la clase `CustomModel` como clase base reutilizable:

```
import os
import re
from ollama import Client
from language_models.base import LanguageModel

class CustomModel(LanguageModel):
    def __init__(self, model):
        super().__init__(type="local")
        self.model = model
        self.client = Client(host=os.getenv("OLLAMA_URL"))

    def prompt_wrapper(self, text: str) -> str:
        messages = [
            {
                "role": "system",
                "content": """You are a data expert...""",
```

```
            },
            {"role": "user", "content": text},
        ]
        return messages

    def generate(self, text: str, retrieved: list) -> str:
        try:
            print("PROMPT", text, flush=True)
            response = self.client.chat(
                model=self.model,
                messages=text,
                keep_alive="10m",
            )
            response_clean = re.sub(
                r"(?<=:\s)([^'\",\s][^,\}]+)",
                r"'\1'",
                response["message"]["content"]
            )
            return self.extract_value_citation(response_clean, retrieved
                )
        except Exception as e:
            return {"status": "fail", "message": str(e)}
```

Esta clase se reutiliza en scripts específicos para cada modelo. Ejemplo para `llama3.1`:

```
from .custom_model import CustomModel

class Llama3_1(CustomModel):
    def __init__(self):
        super().__init__('llama3.1')
```

Cambios en el backend

Se ha corregido y optimizado la función `extract_value_citation`. Esta función se encuentra en `language_models/base.py`. De esta manera mejora el análisis de la salida generada por los modelos.

Adicionalmente, se actualizó la librería `elasticsearch` a la versión `8.7.0` por incompatibilidad con versiones anteriores. El archivo `requirements.txt` quedó de esta manera:

```
python-dotenv
fastapi
pydantic
uvicorn
python-multipart
pandas
sentence-transformers
elasticsearch==8.7.0
qdrant-client
python-multipart
openai==0.28.1
```

```
ollama
anthropic
rerankers
```

Adaptaciones implementadas en análisis de riesgo

Se documentan a continuación las medidas aplicadas en relación con algunos de los riesgos identificados en la sección 4.2. Estas adaptaciones están orientadas al manejo de archivos temporales, control de sesiones, configuración de volúmenes y aislamiento de los modelos utilizados en el sistema.

Control de sesión

Tras evaluar la necesidad de incluir un mecanismo de bloqueo, se determinó que el flujo de ejecución de RetClean ya impide, por diseño, ejecuciones simultáneas sobre el mismo recurso. Esto elimina la necesidad de mecanismos adicionales para evitar concurrencia.

Manejo de logs y archivos temporales

a) Logs de interacción. No se identificó ningún sistema de registro de logs persistentes. La única salida existente es la de los contenedores Docker, que se elimina automáticamente al detenerlos. Esto reduce el riesgo de almacenamiento innecesario de datos sensibles.

b) Eliminación periódica de archivos temporales. Se desarrolló un script para eliminar los archivos temporales generados por el sistema, incluyendo datos cargados por usuarios y resultados. Este script se ejecuta cada 60 minutos, eliminando todo el contenido de los data lakes internos.

```
sudo crontab -e

0 * * * * bash /app/clear_data/clear_temp.sh
```

El script y las instrucciones de uso están organizados en la carpeta `clear_data`, que incluye también un archivo `README.md`.

c) No persistencia de resultados. Para evitar la acumulación de datos entre sesiones, se eliminaron los volúmenes persistentes del archivo `docker-compose.yml`. Las líneas comentadas deshabilitan el almacenamiento:

```
# volumes:
#   - es_storage:/usr/share/elasticsearch/data

# volumes:
#   - qdrant_storage:/qdrant/storage
```

Toda la información generada se mantiene únicamente en memoria durante la sesión activa.

Aislamiento de los modelos

Todos los modelos utilizados por RetClean se agrupan en un único contenedor Docker destinado exclusivamente a la inferencia. Este contenedor no monta volúmenes externos y no tiene acceso directo a archivos ni a la interfaz de la aplicación, lo que refuerza el aislamiento del entorno de ejecución.

4.4. Dataset utilizado

Resumen ejecutivo del conjunto de datos

El conjunto de datos empleado en este trabajo fue recopilado por investigadores del Ankara VM Medical Park Hospital, Turquía, como parte de un estudio clínico prospectivo desarrollado entre junio de 2022 y junio de 2023. Está disponible públicamente desde abril de 2025 en el UCI Machine Learning Repository bajo el nombre Gallstone[19], disponible en: `https://archive.ics.uci.edu/dataset/1150/gallstone`. Su publicación original[3] se encuentra en el artículo *Early prediction of gallstone disease with a machine learning-based method from bioimpedance and laboratory data*[20]. Desde su publicación, ha sido citada en estudios centrados en la detección de enfermedades biliares mediante sistemas CBIR [7], marcos híbridos bayesianos para predicción de riesgo [11] y en investigaciones sobre el papel de la microbiota intestinal en la formación de cálculos [47].

Este conjunto fue creado con el objetivo de predecir la enfermedad de cálculos biliares a partir de datos no invasivos, combinando bioimpedancia, resultados de laboratorio y características demográficas. Se compone de 319 registros de pacientes, 161 de ellos diagnosticados con la enfermedad, lo que garantiza un equilibrio entre clases.

El conjunto está completo, sin valores nulos, y no requiere pasos adicionales de preprocesamiento. Incluye 38 atributos. Entre las variables demográficas figuran la edad, sexo, altura, peso, índice de masa corporal y comorbilidades. Las mediciones de bioimpedancia abarcan agua total, agua extracelular e intracelular, masa magra, masa grasa, masa ósea, grasa visceral, masa muscular visceral, proteína corporal, índice de adiposidad visceral y grado de obesidad. Los datos de laboratorio incluyen niveles de glucosa, colesterol total, HDL, LDL, triglicéridos, AST, ALT, fosfatasa alcalina, creatinina, tasa de filtrado glomerular, proteína C-reactiva, hemoglobina y vitamina D. También incluye una variable binaria denominada Gallstone Status, que señala si el individuo presenta o no la enfermedad.

A continuación, en la tabla 4.2, se presenta la descripción de las variables de este conjunto de datos.

[3] `https://doi.org/10.1097/md.0000000000037258`

Variable	**Tipo**	**Descripción**
Gallstone Status	Binaria	Presencia de cálculos biliares. 0 = No, 1 = Sí
Age	Entera	Edad de la persona. Rango: 20-96 años
Gender	Categórica	Género del paciente. 0 = Hombre, 1 = Mujer
Comorbidity	Categórica	Número de comorbilidades. 0 = Ninguna, 1 = Una, 2 = Dos, 3 = Tres o más
Coronary Artery Disease (CAD)	Binaria	Enfermedad cardiovascular. 0 = No, 1 = Sí
Hypothyroidism	Binaria	Hipotiroidismo. 0 = No, 1 = Sí
Hyperlipidemia	Binaria	Hiperlipidemia. 0 = No, 1 = Sí
Diabetes Mellitus (DM)	Binaria	Diabetes mellitus. 0 = No, 1 = Sí
Weight	Entera	Talla en centímetros
Total Body Water (TBW)	Continua	Agua corporal total (kg)
Extracellular Water (ECW)	Continua	Agua extracelular (kg)
Intracellular Water (ICW)	Continua	Agua intracelular (kg)
Extracellular Fluid/ Total Body Water (ECF/TBW)	Continua	Proporción ECF/TBW (adimensional)
Total Body Fat Ratio (TBFR)	Continua	Porcentaje de grasa corporal total (%)
Lean Mass (LM)	Continua	Masa magra (kg)
Body Protein Content (Protein)	Continua	Contenido proteico corporal (%)
Visceral Fat Rating (VFR)	Entera	Nivel de grasa visceral
Bone Mass (BM)	Continua	Masa ósea (kg)
Muscle Mass (MM)	Continua	Masa muscular (kg)
Obesity	Continua	Grado de obesidad (%)
Total Fat Content (TFC)	Continua	Contenido graso total (kg)
Visceral Fat Area (VFA)	Continua	Área de grasa visceral (cm^2)
Visceral Muscle Area (VMA)	Continua	Área muscular visceral (kg)
Hepatic Fat Accumulation (HFA)	Categórica	Acumulación de grasa hepática. 0 = No, 1-4 = Grados de severidad
Glucose	Continua	Glucosa en sangre (mg/dL)
Total Cholesterol (TC)	Continua	Colesterol total (mg/dL)
Low Density Lipoprotein (LDL)	Continua	Colesterol LDL (mg/dL)
High Density Lipoprotein (HDL)	Continua	Colesterol HDL (mg/dL)
Triglyceride	Continua	Triglicéridos (mg/dL)
Aspartat Aminotransferase (AST)	Continua	Enzima hepática AST (U/L)
Alanin Aminotransferase (ALT)	Continua	Enzima hepática ALT (U/L)
Alkaline Phosphatase (ALP)	Continua	Fosfatasa alcalina (U/L)
Creatinine	Continua	Creatinina (mg/dL)
Glomerular Filtration Rate (GFR)	Continua	Tasa de filtración glomerular (ml/min)
C-Reactive Protein (CRP)	Continua	Proteína C reactiva (mg/L)
Hemoglobin (HGB)	Continua	Hemoglobina (g/dL)
Vitamin D	Continua	Vitamina D (ng/mL)

Tabla 4.2: Características de las variables de Gallstone

4.5. Diseño experimental

Selección de variables de variables objetivos y pivotes

RetClean ejecuta las tareas de limpieza de forma individual para cada variable objetivo, utilizando el resto de atributos como contexto, que actúan como columnas pivote. En cada caso, se introdujeron errores artificiales y se evaluó la capacidad del sistema para restaurar el valor correcto a partir del resto de atributos del paciente.

Para evaluar su capacidad en tareas de imputación y corrección semántica, se seleccionaron variables objetivo que cumplen con los siguientes criterios:

- Presentan valores continuos o categóricos discretos con significado clínico claro
- Están correlacionadas de forma significativa con al menos dos variables adicionales, lo que permite que un sistema basado en recuperación contextual pueda inferir su valor esperado a partir del resto de la instancia
- No son derivables directamente de otras variables, evitando así que la corrección se base únicamente en reglas matemáticas

La selección se realizó tras analizar la matriz de correlación del conjunto de datos original. En la Tabla 4.3 se presentan las variables objetivo seleccionadas junto con aquellas variables cuya correlación supera el valor de 0.50, lo que justifica su elección. Las variables con mayor correlación se utilizaron como pivotes contextuales durante la recuperación.

Variable objetivo	**Tipo**	**Correlaciones más altas**
Visceral Fat Area (VFA)	Continua	TFC (0.87), BMI (0.86), Obesity (0.77), Weight (0.77), TBFR (0.65)
Body Mass Index (BMI)	Continua	TFC (0.89), VFA (0.86), Obesity (0.74), TBFR (0.74), Weight (0.79)
Hepatic Fat Accumulation (HFA)	Categórica ordinal	ALT (0.55), AST (0.51), BMI (0.57), VFA (0.53)

Tabla 4.3: Variables objetivo seleccionadas y sus principales correlaciones

División del conjunto de datos

Se extrajo un 30 por ciento del conjunto original para construir el data lake, utilizado como base de recuperación en el sistema. El 70 por ciento restante se empleó para las evaluaciones experimentales. La partición se realizó de forma estratificada con una semilla fija igual a 123, preservando la proporción original de las variables Gallstone Status y Gender para mantener la representatividad en ambas fracciones.

Simulación de errores y generación de datos corruptos

Para comparar el comportamiento del sistema en condiciones controladas, se utiliza una versión limpia del dataset como punto de partida. A partir de este conjunto, se generan variantes degradadas mediante la introducción de errores simulados. Esta estrategia permite evaluar con precisión la capacidad del sistema para detectar y corregir inconsistencias, midiendo su efectividad frente a datos corruptos de forma reproducible. Este enfoque ha sido adoptado en trabajos anteriores centrados en evaluación de calidad y restauración de datos [34].

La simulación se realiza mediante un script interactivo desarrollado específicamente para este propósito, el cual está disponible en el repositorio de Git, puede ver la estructura de este repositorio en el Apéndice D. A través de este script, es posible seleccionar una columna del dataset original y aplicar transformaciones que emulan errores frecuentes. El sistema identifica el tipo de dato y aplica modificaciones según corresponda.

En columnas categóricas, se reemplazan aleatoriamente ciertos valores por otros válidos de la misma categoría. Esto simula errores de codificación sin introducir categorías nuevas. En columnas numéricas, se generan errores leves mediante alteraciones aritméticas, errores graves mediante outliers, además de vaciado de celdas o sustitución por valores nulos y cadenas vacías.

La proporción y el tipo de error aplicado a cada fila se determinan de forma probabilística, con pesos configurables para representar diferentes niveles de ruido. Esto permite generar datasets sucios reproducibles y ajustables, manteniendo el control sobre las condiciones del experimento.

El archivo resultante se guarda incluyendo el nombre de la columna modificada, lo que facilita su trazabilidad en las fases de evaluación y comparación.

Preparación del entorno de prueba

Las pruebas se ejecutaron sobre una máquina virtual proporcionada por la Escuela de Ingeniería Informática, accesible mediante el host virtual.lab.inf.uva.es y puerto 20201. Las especificaciones completas del sistema, tanto a nivel de hardware como de software, se describen en el Apéndice A.

La herramienta base fue instalada utilizando Docker. Se integró con modelos descargados mediante Ollama y se habilitó el uso de GPU.

Los modelos descargados para este trabajo son: LLaMA 3.1 de 8B, Mistral de 7B y LLaVA de 7B. Se eligieron estos modelos por su disponibilidad abierta, eficiencia en entornos con recursos limitados y capacidad demostrada para tareas de razonamiento estructurado a excepción de LLaMA 3.1 de 8B que se utilizó como modelo por defecto al estar ya integrado en la distribución original de RetClean.

La instalación y uso del entorno se documentan en el Apéndice C que corresponde al manual de usuario. Las modificaciones realizadas al sistema original, incluyendo soporte gráfico y simplificaciones funcionales, se describen en la sección 4.3.

Como parte del flujo de recuperación de RetClean, se construyó un data lake indexado utilizando el 30 por ciento del conjunto de datos. Esta funcionalidad está integrada en el sistema y permite realizar búsquedas semánticas sobre el repositorio para asistir en la imputación de valores [39].

Configuraciones a evaluar

A partir de las variables objetivo y los pivotes contextuales seleccionados y definidos en la sección 4.5, se aplicó recuperación semántica en cada caso para mejorar la precisión de las respuestas generadas. Esta funcionalidad, integrada como una opción en RetClean, permite reforzar el contexto con relaciones estructuradas previamente definidas, sin requerir que el conjunto de datos provenga del sistema. El uso de recuperación semántica está respaldado por resultados recientes en entornos médicos, donde la integración de conocimiento semántico mostró mejoras significativas en tareas de razonamiento clínico [18].

Las configuraciones resultantes se resumen en la Tabla 4.4. Estas configuraciones se ejecutaron con los tres modelos (*on-premises*) disponibles.

Target	**Tipo**	**Pivotes contextuales**	**Recuperación**
VFA	Continua	TFC, BMI, Obesity, Weight, TBFR	Semántica
BMI	Continua	TFC, VFA, Obesity, TBFR, Weight	Semántica
HFA	Categórica ordinal	ALT, AST, BMI, VFA	Semántica

Tabla 4.4: Configuraciones evaluadas por variable objetivo

Aunque las tres variables objetivo fueron evaluadas bajo el mismo marco, cada una corresponde a una tarea diferente. En BMI y VFA se predicen valores continuos, mientras que en HFA se realiza una clasificación ordinal. El análisis es descriptivo y no se aplicaron comparaciones estadísticas entre ellas, ya que presentan escalas distintas y estructuras clínicas no equivalentes.

Métricas de evaluación

La evaluación del rendimiento se realizó considerando cada predicción generada por el sistema sobre las celdas sucias simuladas. Se definieron cuatro posibles casos para determinar verdaderos positivos, falsos positivos, falsos negativos y verdaderos negativos:

- **TP**: El modelo corrigió el valor sucio y la predicción coincide con el valor limpio.

- **FP**: El modelo modificó el valor sucio, pero la predicción no coincide con el valor limpio.
- **FN**: El valor sucio era incorrecto, pero el modelo no realizó ninguna corrección.
- **TN**: El valor era correcto y el modelo lo dejó sin cambios.

Estas categorías se aplican a las celdas modificadas por el sistema, a excepción del TN, y se comparan con las versiones limpias correspondientes. Para evaluar la calidad de las predicciones, se utilizaron las métricas resumidas en la Tabla 4.5.

En el caso de variables continuas, se consideró que una predicción constituye un verdadero positivo si el error relativo respecto al valor limpio es menor al 5 %. Este criterio se aplicó solo para la asignación de verdaderos positivos.

Métrica	**Aplicación**	**Fórmula**
Precisión	Categórica y numérica	$\frac{TP}{TP+FP}$
Recall	Categórica y numérica	$\frac{TP}{TP+FN}$
F1-score	Categórica y numérica	$F_1 = 2 \cdot \frac{P \cdot R}{P+R}$
Exact match	Categórica y numérica	$\frac{\text{aciertos exactos}}{\text{total de filas}}$
Error absoluto	Numérica	$\lvert y_{\text{pred}} - y_{\text{real}} \rvert$
Error relativo	Numérica	$\frac{\lvert y_{\text{pred}} - y_{\text{real}} \rvert}{\lvert y_{\text{real}} \rvert}$

Tabla 4.5: Métricas utilizadas para la evaluación de las predicciones

Para calcular estas métricas se desarrolló un script que procesa las salidas generadas por RetClean y las compara con los datos sucios y los valores originales. A partir de esta comparación, se realizan los cálculos necesarios y se genera un reporte con los resultados obtenidos. El script se encuentra disponible en el repositorio de Git.

Capítulo 5

Resultados y Discusión

5.1. Resultados obtenido de las configuraciones ejecutadas

Comparación de rendimiento computacional entre modelos

Se evaluaron los tres modelos disponibles ejecutados con GPU para comparar su eficiencia relativa al limpiar cada variable objetivo. RetClean ejecuta una tarea de limpieza independiente por cada columna seleccionada como target, utilizando el resto de atributos como contexto. La selección de las variables objetivo HFA, BMI y VFA se detalló en la Sección 4.5. Los tiempos registrados corresponden a cada una de esas ejecuciones.

Los resultados se presentan en la tabla 5.1 como análisis descriptivo. No se aplicaron pruebas estadísticas formales para determinar diferencias significativas entre modelos.

Modelo	**Tiempo de ejecución (ms)**			**Memoria usada (MB)**
	HFA	**BMI**	**VFA**	
Llama3.1	652.721	1106.539	1099.197	6395
Mistral	529.374	871.697	868.427	5825
Llava	639.896	948.937	914.180	6476

Tabla 5.1: Tiempos de ejecución por variable y memoria usada por modelo

El modelo Mistral aplicado a la variable HFA fue el que presentó el menor tiempo de ejecución en entorno GPU. A partir de este caso se realizó una comparación directa entre su comportamiento en GPU y CPU para evaluar el impacto de la aceleración por hardware gráfico. En CPU, el modelo tardó 9764 segundos, equivalente a 162 minutos, mientras que en GPU el tiempo se redujo a 529 segundos, es decir, 8,8 minutos. La aceleración representa una mejora del 94,6 % en tiempo de procesamiento.

Métrica	CPU	GPU
Tiempo de ejecución (s)	9764.54	529.37

Tabla 5.2: Comparación del tiempo de ejecución del modelo Mistral en CPU y GPU

Evaluación de resultados por variable objetivo

Como primer paso en la evaluación, se presenta una tabla resumen de la matriz de confusión para cada combinación de modelo y variable objetivo. Esta matriz permite observar el comportamiento detallado del sistema en términos de verdaderos positivos (TP), falsos positivos (FP), falsos negativos (FN) y verdaderos negativos (TN).

Las definiciones empleadas corresponden a las descritas en la sección4.5, donde:

- **TP (verdaderos positivos)**: el modelo corrigió correctamente un valor alterado.
- **FP (falsos positivos)**: el modelo modificó un valor que era correcto.
- **FN (falsos negativos)**: el modelo no corrigió un valor alterado.
- **TN (verdaderos negativos)**: el modelo dejó sin cambios un valor que era correcto.

La tabla 5.3 presenta los resultados obtenidos en cada caso.

Modelo	Variable	TP	FP	FN	TN	Total
LLaVA	HFA	69	89	40	26	224
LLaVA	BMI	16	54	152	2	224
LLaVA	VFA	5	60	158	1	224
Mistral	HFA	66	92	37	29	224
Mistral	BMI	18	52	153	1	224
Mistral	VFA	6	59	159	0	224
LLaMA 3.1	HFA	60	98	43	23	224
LLaMA 3.1	BMI	17	53	153	1	224
LLaMA 3.1	VFA	8	57	157	2	224

Tabla 5.3: Matriz de confusión por modelo y variable objetivo

En función de estos valores, se calcularon posteriormente las métricas agregadas: precisión, recall, F1-score, exact match, error absoluto y error relativo. Los resultados obtenidos se presentan en la tabla 5.4, que resume el rendimiento de los tres modelos evaluados sobre las variables objetivo definidas: Visceral Fat Area (VFA), Body Mass Index (BMI) y Hepatic Fat Accumulation (HFA). Cada configuración utilizó recuperación semántica y un conjunto de pivotes contextuales, tal como se describió en la sección 4.5.

Métrica	**VFA**			**BMI**			**HFA**		
	LLaMA	Mistral	LLaVA	LLaMA	Mistral	LLaVA	LLaMA	Mistral	LLaVA
Precisión	0.1231	0.0923	0.0769	0.2429	0.2571	0.2286	0.3797	0.4177	0.4367
Recall	0.0485	0.0364	0.0307	0.1000	0.1053	0.0952	0.5825	0.6408	0.6330
F1-score	0.0696	0.0522	0.0439	0.1417	0.1494	0.1345	0.4598	0.5057	0.5169
Exact match	0.0134	0.0000	0.0089	0.0134	0.0045	0.0089	0.3705	0.4241	0.4241
Error absoluto	3.4582	3.0583	5.4455	3.7670	3.9318	4.4019	-	-	-
Error relativo	0.3019	0.3361	0.4972	0.1332	0.1283	0.1595	-	-	-

Tabla 5.4: Resultados de evaluación para cada modelo y variable objetivo

Los resultados muestran diferencias notables según el tipo de variable. En las variables continuas (VFA y BMI), los valores de precisión, recall y F1-score son bajos en todos los modelos, con un rendimiento máximo de F1 igual a 0.1494 en el caso de BMI usando Mistral. El error relativo promedio se mantiene entre 0.1283 y 0.4972, siendo LLaVA el modelo con mayor desviación respecto al valor real en ambos targets continuos.

En cambio, la variable categórica ordinal HFA presentó valores significativamente más altos en todas las métricas. El modelo LLaVA obtuvo la mejor F1 (0.5169), seguido por Mistral (0.5057) y LLaMA (0.4598). Los valores de exact match alcanzan 0.4241 en los dos mejores modelos, lo que indica una mayor capacidad para predecir correctamente el valor exacto de esta variable cuando se trata de categorías discretas.

5.2. Análisis funcional

Se seleccionó la variable BMI procesada con el modelo Mistral, dado que obtuvo el menor error relativo medio entre las variables continuas. El análisis se realiza fila a fila, contrastando el valor original, el dato sucio y la predicción generada.

En total se procesaron 224 observaciones. Según la matriz de confusión mostrada en la tabla 5.3, se identificaron 18 verdaderos positivos (TP), 52 falsos positivos (FP), 153 falsos negativos (FN) y 1 verdadero negativo (TN). La mayoría de los errores corresponden a falsos negativos, lo que refleja una baja sensibilidad en la detección de valores alterados para esta variable. Aun así, el modelo corrigió correctamente 18 valores.

Al examinar casos específicos, se observa que el modelo suele repetir el valor sucio sin aplicar correcciones, incluso cuando el error es leve. Por ejemplo, si el valor original era 25.4 y el dato sucio 40.9, la predicción también fue 40.9, sin modificar el valor. Un comportamiento similar se repitió con el valor original 27.49, donde tampoco se detectó el error. En contraste, también se identificaron predicciones correctas. En una observación con dato sucio 32.2 y valor original 30.2, el modelo generó correctamente 30.2. En otro caso, el valor sucio era 28.2 y la predicción restauró el original 25.4. Estos ejemplos reflejan

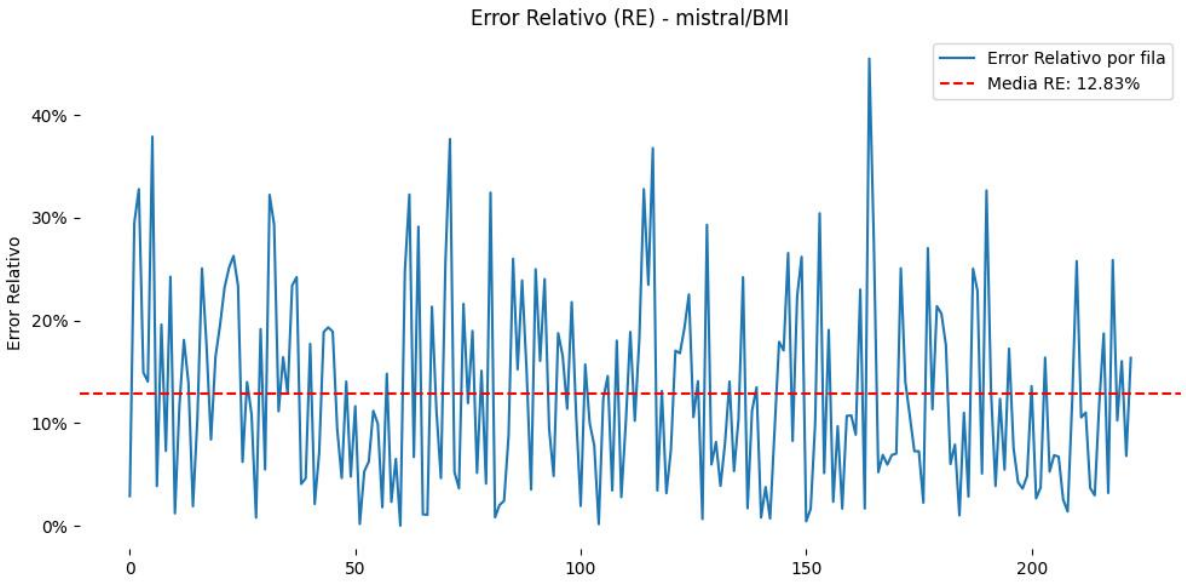

Figura 5.1: Error relativo fila a fila en BMI con Mistral

que, aunque la detección de errores es limitada, el modelo puede recuperar valores exactos en contextos particulares.

Sin embargo, no se detectan patrones consistentes. Muchos errores leves no se corrigen, mientras que algunas distorsiones moderadas sí lo son. Esto sugiere que la capacidad de restauración depende del contexto específico de cada observación más que de la magnitud del error. Para observar la magnitud y distribución de los errores, la figura 5.1 muestra el error relativo fila a fila. La línea azul representa el valor de error relativo para cada predicción y la línea roja indica el promedio general. Aunque no se registran picos extremos, los errores se mantienen dispersos y varios casos superan el 20 por ciento.

5.3. Discusión técnica

Los resultados deben interpretarse considerando las limitaciones descritas en la sección 4.1. Como se expone allí, la versión actual de RetClean no incluye el modelo fine-tuned ni los filtros de recuperación utilizados en el estudio original [39], lo que impidió replicar ese experimento. En su lugar, se empleó la configuración funcional disponible en el repositorio oficial, validada con un dataset reducido.

La evaluación se realizó sobre el conjunto de datos descrito en la sección 4.4, completo y sin valores faltantes [19]. Este conjunto actúa como *ground-truth*, lo que permite comparar directamente los valores restaurados con los valores reales [12]. En este contexto, se observa que el rendimiento de los modelos evaluados para corregir o restaurar valores corruptos depende significativamente del tipo de variable y de la estructura semántica utilizada

Variables continuas

En las tareas sobre variables continuas, los tres modelos muestran un rendimiento limitado. Los valores bajos de F1-score y exact match indican que, aunque algunos valores pueden acercarse al real, la predicción rara vez coincide con el valor limpio original. El error absoluto promedio para VFA se mantiene por encima de 3 unidades, y para BMI cerca de 4. Esto limita la utilidad de las predicciones para tareas donde se requiere precisión cuantitativa. La inclusión de recuperación semántica no fue suficiente para compensar la variabilidad en estas variables.

El modelo Mistral obtuvo ligeramente mejores resultados que LLaMA y LLaVA en BMI, tanto en F1 como en error relativo. No obstante, la diferencia no es sustancial. LLaVA mostró el peor rendimiento global en variables continuas, especialmente en VFA, con el mayor error relativo (0.4972), lo que indica una sensibilidad alta a valores atípicos o distorsiones en los datos de entrada.

Variable categórica ordinal

En contraste, la predicción de la variable HFA muestra una mejora clara. La clasificación ordinal favorece la tarea de inferencia contextual, ya que la recuperación semántica permite al modelo identificar patrones clínicos más estables entre atributos como ALT, AST, VFA y BMI. Aquí, los modelos generativos sí logran restaurar el valor original con mayor eficacia, alcanzando niveles de F1 comparables a los obtenidos en sistemas supervisados básicos.

El uso de recuperación semántica parece tener mayor impacto cuando se trabaja con variables categóricas, especialmente si están respaldadas por pivotes contextuales bien definidos. Esto coincide con los planteamientos previos en la literatura revisada, donde se sugiere que el razonamiento estructurado es más eficaz cuando las relaciones entre variables son discretas y coherentes.

Comparación entre modelos

LLaVA mostró el mejor comportamiento en la predicción de HFA, mientras que Mistral se posicionó como el más balanceado en tareas continuas. LLaMA, aunque funcional como modelo por defecto, presentó rendimiento inferior en la mayoría de los escenarios. Las diferencias de rendimiento no justifican un modelo como dominante, pero permiten identificar el tipo de tarea donde cada uno ofrece mejor comportamiento.

El uso de GPU fue necesario para mantener los tiempos de ejecución dentro de márgenes operativos razonables. Las diferencias de consumo y latencia se detallan en la sección 5.1. Aunque LLaVA tuvo un mayor consumo de memoria, su desempeño en tareas categóricas puede justificar su uso en contextos donde se prioriza exactitud sobre eficiencia.

Capítulo 6

Conclusiones y trabajos futuros

En este trabajo se aplicaron y evaluaron modelos generativos de lenguaje de gran tamaño (LLMs) en tareas de mejora de la calidad de datos estructurados mediante la adaptación y extensión del sistema RetClean. Los resultados obtenidos proporcionan conclusiones relevantes tanto desde el punto de vista teórico como práctico. A partir de la revisión realizada, se identificaron múltiples enfoques recientes desarrollados entre 2023 y 2025 que integran modelos generativos con técnicas de recuperación aumentada, también conocida como RAG, para optimizar tareas de limpieza en datos tabulares, semiestructurados y en dominios como el clínico, legal y administrativo. Esto evidencia un creciente interés en automatizar la calidad de datos mediante inteligencia artificial generativa.

Los experimentos realizados muestran que los LLMs pueden asistir eficazmente en tareas como detección y corrección de errores sintácticos y semánticos, imputación de valores faltantes y sugerencias de transformación, especialmente en variables categóricas u ordinales. Sin embargo, el desempeño sobre variables continuas fue limitado en términos de precisión exacta, lo que representa un desafío adicional en este tipo de datos. Comparados con técnicas tradicionales, los modelos generativos ofrecen mayor flexibilidad y capacidad contextual, facilitando el tratamiento de datos heterogéneos y no estructurados. No obstante, presentan restricciones importantes como el elevado consumo computacional y la necesidad de una ingeniería de prompts adecuada para garantizar su efectividad.

En cuanto a su adaptabilidad, la arquitectura modular de RetClean permitió integrar modelos genéricos y personalizados ejecutados localmente, lo que facilitó la configuración contextual a través de recuperación semántica con variables relevantes. Esta flexibilidad es una ventaja clara, aunque la adaptación en variables continuas y tipos de datos diversos requiere mejoras adicionales, sobre todo en mecanismos de evaluación semántica y validación automatizada. En lo relativo a la privacidad, el uso de modelos locales en entornos controlados contribuyó a minimizar los riesgos derivados de la transferencia y almacenamiento de datos externos. A pesar de ello, se identificaron vulnerabilidades como

la persistencia temporal de archivos, la carencia de validación posterior a la inferencia y la ausencia de mecanismos de anonimización en registros de sistema.

Respecto al cumplimiento de objetivos, se alcanzó el propósito principal del trabajo mediante la aplicación y evaluación de modelos generativos para mejorar la calidad de datos estructurados, logrando avances en varios objetivos específicos. Se realizó una revisión sistemática de literatura científica y técnica para identificar metodologías y herramientas emergentes basadas en LLMs. Además, se analizó a fondo la plataforma RetClean, evaluando sus capacidades y limitaciones. Se documentaron riesgos concretos de privacidad y seguridad, orientando medidas mitigadoras. Se integraron modelos generativos locales en el flujo de trabajo, mejorando su configuración, persistencia y uso combinado con recuperación semántica. Finalmente, se evaluó la capacidad del sistema para identificar, corregir y sugerir mejoras en datos tabulares, utilizando un dataset clínico con errores simulados, y se compararon los resultados empleando métricas cuantitativas como precisión, recall, F1-score, exactitud y errores absolutos y relativos.

En términos de resultados, se observó que los modelos generativos generan valores plausibles, pero con baja coincidencia exacta respecto al dato original. Las tasas de exact match fueron inferiores al 20 % en la mayoría de variables. En particular, el modelo Mistral logró los errores relativos más bajos en la variable BMI, aunque con un número elevado de falsos positivos. El análisis fila a fila reveló errores frecuentes, aunque no extremos, y evidenció que los modelos tienden a modificar también registros originalmente correctos, lo cual afecta negativamente a la precisión global del sistema. El uso de métricas como el error absoluto y relativo permitió una evaluación más detallada del comportamiento de los modelos sobre variables continuas, donde la coincidencia literal no siempre es un indicador útil de calidad. Asimismo, el uso de modelos locales, desplegados en contenedores y sin transmisión de datos externos, constituye una solución viable para entornos donde la privacidad resulta prioritaria.

A partir de los hallazgos obtenidos, se proponen diversas líneas de trabajo futuro. Entre ellas, se plantea ajustar y optimizar los prompts para mejorar la instrucción de restauración y evitar modificaciones indebidas; evaluar modelos generativos con mayor número de parámetros y capacidad contextual; probar el sistema con datasets más amplios y complejos que permitan validar su escalabilidad; e incorporar variables textuales con el objetivo de analizar el potencial de los LLMs en la restauración semántica más allá de los datos numéricos.

Bibliografía

[1] Ali, M., Arunasalam, A., and Farrukh, H. Understanding users' security and privacy concerns and attitudes towards conversational AI platforms. In *2025 IEEE Symposium on Security and Privacy (SP)* (2025), pp. 298–316.

[2] Atkinson-Abutridy, J. *Grandes Modelos de Lenguaje: Conceptos, Técnicas y Aplicaciones.*, 1st ed. ed. Marcombo, S.A., Barcelona, 2023.

[3] Azeroual, O. Can Generative AI transform Data Quality? a critical discussion of ChatGPT's capabilities. *Academia Engineering 1*, 4 (2024).

[4] Batini, C., Ceri, S., and Navathe, S. B. *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings, 1992.

[5] Batini, C., and Scannapieca, M. *Data quality : concepts, methodologies and techniques*, 1st ed. 2006. ed. Data-centric systems and applications. Springer, Berlin ;, 2006.

[6] Bengesi, S., El-Sayed, H., Sarker, M. K., Houkpati, Y., Irungu, J., and Oladunni, T. Advancements in Generative AI: A comprehensive review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers. *IEEE Access 12* (2024), 69812–69837.

[7] Bozdağ, A., Yildirim, M., Karaduman, M., Mutlu, H. B., Karaduman, G., and Aksoy, A. Detection of Gallbladder Disease Types using a Feature Engineering-based developed CBIR System. *Diagnostics 15* (2025).

[8] Caballero Munoz-Reja, I. *Calidad de datos*. Informatica. Ediciones de la U, Bogota, 2019.

[9] Carrera-Rivera, A., Ochoa, W., Larrinaga, F., and Lasa, G. How-to conduct a systematic literature review: A quick guide for computer science research. *MethodsX 9* (2022), 101895.

[10] Cervo, D., and Allen, M. *Master data management in practice: achieving true customer MDM*, 1st ed. ed., vol. 559 of *Wiley Corporate F and A*. Wiley, Newark, 2011.

[11] Chakraborty, C., and Mukherjee, N. Bayesian Hybrid Machine Learning of Gallstone Risk, 2025.

[12] Christen, P. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, 1st ed. 2012. ed. Data-centric systems and applications. Springer-Verlag, Berlin ;, 2012.

[13] Creative Commons. Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). https://creativecommons.org/licenses/by-nc/4.0/, 2013. Fecha de acceso: 2025-07-10.

[14] DAMA España. Código de calidad de la información. https://www.dama-nl.org/wp-content/uploads/2020/09/C%C3%B3digo-de-calidad-de-la-informaci%C3%B3n-2019-DAMA-ES.pdf, 2019. Fecha de acceso: 2025-07-17.

[15] Datos.gob.es. Rag (retrieval-augmented generation), la llave que abre la puerta de la precisión en los modelos del futuro. https://datos.gob.es/es/blog/rag-retrieval-augmented-generation-la-llave-que-abre-la-puerta, 2023. Fecha de acceso: 2025-07-17.

[16] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805, 2018. Fecha de acceso: 2025-07-17.

[17] Docker, Inc. Install Docker Engine on Ubuntu. https://docs.docker.com/engine/install/ubuntu/, may 2025. Fecha de acceso: 2025-07-01.

[18] Elkin, P. L., Mehta, G., LeHouillier, F., Resnick, M., Mullin, S., Tomlin, C., Resendez, S., Liu, J., Nebeker, J. R., and Brown, S. H. Semantic clinical Artificial Intelligence vs native Large Language Model performance on the USMLE. *JAMA Network Open 8*, 4 (Apr 2025), e256359.

[19] Esen, I., Arslan, H., Aktürk, S., Gülşen, M., Kültekin, N., and Özdemir, O. Gallstone. UCI Machine Learning Repository, 2024. DOI: https://doi.org/10.1097/md.0000000000037258.

[20] Esen, I., Arslan, H., Esen, S. A., Gülşen, M., Kültekin, N., and Özdemir, O. Early prediction of Gallstone Disease with a Machine Learning-based method from Bioimpedance and laboratory data. *Medicine 103*, 8 (2024).

[21] European Data Protection Board. AI privacy risks & mitigations – Large Language Models (LLMs). https://www.edpb.europa.eu/system/files/2025-04/ai-privacy-risks-and-mitigations-in-llms.pdf, Mar. 2025. Fecha de acceso: 2025-07-07.

[22] Frees, E. Loss data analytics, 2018.

[23] Fu, Q., and Easton, J. M. Understanding Data Quality: Ensuring Data Quality by design in the rail industry. In *2017 IEEE International Conference on Big Data (Big Data)* (2017), pp. 3792–3799.

[24] Geroimenko, V. *The Essential Guide to Prompt Engineering : Key Principles, Techniques, Challenges, and Security Risks*, 1st ed. 2025. ed. SpringerBriefs in Computer Science. Springer Nature Switzerland, Cham, 2025.

[25] Gorelik, A. Architecting the Data Lake. In *The Enterprise Big Data Lake*. O'Reilly Media, Incorporated, United States, 2019.

[26] Han, J., Kamber, M., and Pei, J. *Data mining : concepts and techniques*, 3rd ed. ed. The Morgan Kaufmann series in data management systems. Elsevier, Burlington, Mass, 2012.

[27] ISO/IEC. ISO/IEC 25012:2008 - Software Engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model. https://www.iso.org/standard/35746.html, 2008. Fecha de acceso: 2025-07-17.

[28] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. https://arxiv.org/abs/2310.06825, 2023. Fecha de acceso: 2025-07-07.

[29] Kamath, U., Keenan, K., Somers, G., and Sorenson, S. *Large Language Models: A Deep Dive — Bridging Theory and Practice*, 1st ed. 2024. ed. Springer Nature Switzerland, Cham, 2024.

[30] Karimi, S., Rasel, A. A., and Abdullah, M. S. Non-English Natural Language Interface to Databases: A systematic review. In *2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (2022), pp. 0391–0397.

[31] Kim, D., Won, H., Gil, M.-S., and Moon, Y.-S. Panacea: An automatic Data Migration Framework for constructing Internet-Scale Open Data Lakes. *Software: Practice and Experience 55*, 6 (2025), 1106–1126.

[32] Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., and Linkman, S. Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology 51*, 1 (2009), 7–15. Special Section - Most Cited Articles in 2002 and Regular Research Papers.

[33] Lakatos, R., Urbán, E. K., Szabó, Z. J., Pozsga, J., Csernai, E., and Hajdu, A. Designing Prompts and creating Cleaned Scientific Text for Retrieval Augmented Generation for more precise responses from Generative Large Language

Models. In *2024 IEEE 3rd Conference on Information Technology and Data Science (CITDS)* (2024), pp. 1–6.

[34] Li, P., Rao, X., Blase, J., Zhang, Y., Chu, X., and Zhang, C. Cleanml: A study for evaluating the impact of data cleaning on ml classification tasks. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2021), pp. 13–24.

[35] Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual Instruction Tuning. https://arxiv.org/abs/2304.08485, 2023. Fecha de acceso: 2025-07-10.

[36] Lopez Porrero, B. E., e libro, C., and Perez Vazquez, R. A. *Limpieza de datos*. Editorial Feijóo, Santa Clara, 2009.

[37] Marcel, R. V. P., Fernando, B. E. M., and Roberto, Y. V. J. A brief history of the artificial intelligence: ChatGPT : The evolution of GPT. In *2023 18th Iberian Conference on Information Systems and Technologies (CISTI)* (2023), pp. 1–5.

[38] McGilvray, D. *Executing Data Quality projects : ten steps to quality data and trusted information*, 2nd ed. ed. Academic Press, London, England, 2021.

[39] Naeem, Z. A., Ahmad, M. S., Eltabakh, M., Ouzzani, M., and Tang, N. Retclean: Retrieval-based data cleaning using LLMs and Data Lake. *Proc. VLDB Endow. 17*, 12 (Aug. 2024), 4421–4424.

[40] NVIDIA Corporation. Installing the NVIDIA container toolkit. https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html#ubuntu, 2025. Fecha de acceso: 2025-07-01.

[41] OpenAI, Achiam, J., Adler, S., Agarwal, S., ..., and Zoph, B. GPT-4 Technical Report. https://arxiv.org/abs/2303.08774, 2023. Fecha de acceso: 2025-07-17.

[42] Parlamento Europeo y Consejo de la Unión Europea. Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo de 27 de abril de 2016 relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento General de Protección de Datos). https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX:02016R0679-20160504, 2016. Fecha de acceso: 2025-07-07.

[43] Project Management Institute (Upper Darby, P. *Guía de los fundamentos para la dirección de proyectos : (Guía del PMBOK)*, 6ª ed. ed. Project Management Institute, Pennsylvania, 2017.

[44] Pérez-Fuentes, M. d. C. *Manual práctico para la realización de una revisión sistemática*. UNIVERSIDAD DE ALMERÍA, 2023.

[45] Rahm, E., and Do, H. H. Data cleaning: Problems and current approaches. https://dbs.uni-leipzig.de/research/publications/data-cleaning-problems-and-current-approaches, 2000. Fecha de acceso: 2025-07-17.

[46] Shi, X., Wang, J., Chung, G. J., Julian, D., and Qiao, L. Data imputation based on Retrieval-Augmented Generation. *Applied Sciences 15*, 13 (2025).

[47] Tan, L., Jia, F., and Liu, Y. Advances in research on the role of Gut Microbiota in the pathogenesis and precision management of Gallstone Disease. *Frontiers in Medicine 12* (2025).

[48] Tang, X., Liu, W., Wu, S., Yao, C., Yuan, G., Ying, S., and Chen, G. Queryartisan: Generating data manipulation codes for Ad-hoc analysis in Data Lake. *Proc. VLDB Endow. 18*, 2 (Oct. 2024), 108–116.

[49] Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, and et al. LaMDA: Language Models for Dialog Applications. https://arxiv.org/abs/2201.08239, 2022. Fecha de acceso: 2025-07-17.

[50] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and Efficient Foundation Language Models. https://arxiv.org/abs/2302.13971, 2023. Fecha de acceso: 2025-07-17.

[51] Ubuntu Community. Installation on low memory systems. https://help.ubuntu.com/community/Installation/LowMemorySystems. Fecha de acceso: 2025-07-01.

[52] Ubuntu Manuals. `links` - lynx-like alternative character mode WWW browser. https://manpages.ubuntu.com/manpages/jammy/man1/links2.1.html. Fecha de acceso: 2025-07-01.

[53] Ubuntu Packages. Links2 2.25-1build1 amd64.deb for Ubuntu 22.04 lts. https://ubuntu.pkgs.org/22.04/ubuntu-universe-amd64/links2_2.25-1build1_amd64.deb.html. Fecha de acceso: 2025-07-01.

[54] Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., and Tang, Y. A brief overview of ChatGPT: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica 10*, 5 (2023), 1122–1136.

[55] Yen, A. R., Kapustin, C. V., Burga-Durango, D., Tello-Saenz, C. A., Guarda, T., Portela, F., and Gatica, G. Automated system for improving audit data processing through DAMA-DMBOK best practices and Low-Code. In *Advanced Research in Technologies, Information, Innovation and Sustainability*, Communications in Computer and Information Science. Springer Nature Switzerland, Cham, 2025, pp. 434–444.

Estudios primarios

[BADG25] Fabian Biester, Mohamed Abdelaal, and Daniel Del Gaudio. Llmclean: Context-aware tabular data cleaning via llm-generated ofds. In Joe Tekli, Johann Gamper, Richard Chbeir, Yannis Manolopoulos, Salma Sassi, Mirjana Ivanovic, Genoveva Vargas-Solar, and Ester Zumpano, editors, *New Trends in Database and Information Systems*, pages 68–78, Cham, 2025. Springer Nature Switzerland.

[BDVI24] Divya Biradar, Rahul Dattangire, Ruchika Vaidya, and NagaSuryaShivani Inti. Improving the quality of diabetic data with large language model-driven cleaning techniques. In *2024 International Conference on Intelligent Systems and Advanced Applications, ICISAA 2024*, 2024.

[BLD+23] Quinten Bolding, Baohao Liao, Brandon Denis, Jun Luo, and Christof Monz. Ask language model to clean your noisy translation data. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3215–3236, Singapore, December 2023. Association for Computational Linguistics.

[BvdLN+23] Nikolay Bogoychev, Jelmer van der Linde, Graeme Nail, Barry Haddow, Jaume Zaragoza-Bernabeu, Gema Ramírez-Sánchez, Lukas Weymann, Tudor Nicolae Mateiu, Jindřich Helcl, and Mikko Aulamo. Opuscleaner and opustrainer, open source toolkits for training machine translation and large language models, 2023.

[CLW+24] Zhou Chen, Ming Lin, Zimeng Wang, Mingrun Zang, and Yuqi Bai and. Preparedllm: effective pre-pretraining framework for domain-specific large language models. *Big Earth Data*, 8(4):649–672, 2024.

[CTFL23] Chengliang Chai, Nan Tang, Ju Fan, and Yuyu Luo. Demystifying artificial intelligence for data preparation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, page 13 – 20, 2023.

[EY24] Asım Ersoy and Olcay Taner Yıldız. Organic data-driven approach for turkish grammatical error correction and llms, 2024.

[GGBR24] Skander Ghazzai, Daniela Grigori, Boualem Benatallah, and Raja Rebai. Harnessing gpt for data transformation tasks. In *2024 IEEE International Conference on Web Services (ICWS)*, pages 1329–1334, 2024.

[HSL+24] Xiaobao Huang, Mihir Surve, Yuhan Liu, Tengfei Luo, Olaf Wiest, Xiangliang Zhang, and Nitesh V. Chawla. Application of large language models in chemistry reaction data extraction and cleaning. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, page 3797–3801, New York, NY, USA, 2024. Association for Computing Machinery.

[LFT24] Lan Li, Liri Fang, and Vetle I. Torvik. Autodcworkflow: Llm-based data cleaning workflow auto-generation and benchmark, 2024.

[LLT25] Chunhua Liu, Hong Yi Lin, and Patanamon Thongtanunam. Too noisy to learn: Enhancing data quality for code review comment generation, 2025.

[LQL+25] Hao Liang, Meiyi Qiang, Yuying Li, Zefeng He, Yongzhen Guo, Zhengzhou Zhu, Wentao Zhang, and Bin Cui. Mathclean: A benchmark for synthetic mathematical data cleaning, 2025.

[MAD+24] Manuel Mondal, Julien Audiffren, Ljiljana Dolamic, Gérôme Bovet, and Philippe Cudré-Mauroux. Cleaning semi-structured errors in open data using large language models. In *2024 11th IEEE Swiss Conference on Data Science (SDS)*, pages 258–261, 2024.

[MPSD25] Elyas Meguellati, Nardiena Pratama, Shazia Sadiq, and Gianluca Demartini. Are large language models good data preprocessors?, 2025.

[MR24] Pavitra Mehra and El Kindi Rezig. Leveraging structured and unstructured data for tabular data cleaning. In *2024 IEEE International Conference on Big Data (BigData)*, pages 5765–5768, 2024.

[NAE+24] Zan Ahmad Naeem, Mohammad Shahmeer Ahmad, Mohamed Eltabakh, Mourad Ouzzani, and Nan Tang. Retclean: Retrieval-based data cleaning using llms and data lakes. *Proc. VLDB Endow.*, 17(12):4421–4424, August 2024.

[NdTB25] Rihem Nasfi, Guy de Tré, and Antoon Bronselaer. Improving data cleaning by learning from unstructured textual data. *IEEE Access*, 13:36470 – 36491, 2025.

[NZM+24] Wei Ni, Kaihang Zhang, Xiaoye Miao, Xiangyu Zhao, Yangyang Wu, and Jianwei Yin. Iterclean: An iterative data cleaning framework with large

language models. In *ACM International Conference Proceeding Series*, page 100 – 105, 2024.

[PSA+24] Emanuele Pucci, Camilla Sancricca, Salvatore Andolina, Cinzia Cappiello, Maristella Matera, and Anna Barberio. Improving understandability and control indata preparation: A human-centered approach. In Giancarlo Guizzardi, Flavia Santoro, Haralambos Mouratidis, and Pnina Soffer, editors, *Advanced Information Systems Engineering*, pages 284–299, Cham, 2024. Springer Nature Switzerland.

[SLW+25] Yingli Shen, Wen Lai, Shuo Wang, Xueren Zhang, Kangyang Luo, Alexander Fraser, and Maosong Sun. Dcad-2000: A multilingual dataset across 2000+ languages with data cleaning as anomaly detection, 2025.

[WLW23] Runhui Wang, Yuliang Li, and Jin Wang. Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1502–1515, 2023.

[ZHW24] Shuo Zhang, Zezhou Huang, and Eugene Wu. Data cleaning using large language models, 2024.

[ZL24] Mehdi Zaeifi and Beiyu Lin. Smartphone usage data cleaning using llm-based processing. In *2024 IEEE International Conference on Big Data (BigData)*, pages 8871–8873, 2024.

[ZNS+24] Samuel Zuehlke, Joel Nitu, Simone Sandler, Oliver Krauss, and Andreas Stockl. Self-repairing data scraping for websites. In *International Conference on Electrical, Computer, Communications and Mechatronics Engineering, ICECCME 2024*, 2024.

Apéndices

Apéndice A

Plan de Proyecto y Ejecución

Este apéndice presenta el plan de trabajo definido al inicio del proyecto y cómo se desarrolló en la práctica. La planificación se elaboró antes del 27 de febrero de 2025, fecha oficial de inicio. Las actividades se dividieron por fases, con fechas de inicio y duración estimada para cada una. Se utilizó un diagrama de Gantt para organizar y hacer seguimiento del proyecto. La herramienta muestra de forma visual las tareas previstas, su duración y la distribución temporal. Es una herramienta común en gestión de cronogramas, recomendada por el Project Management Institute como parte de los procesos de planificación [43].

A.1. Planificación del trabajo

Se estimó una dedicación de 10 horas por semana. La fecha prevista de finalización fue el 19 de junio de 2025. En esta etapa no se contemplaron modificaciones técnicas ni ajustes en el entorno, ya que aún no se había definido el caso de estudio ni estaba previsto si se trabajaría con una herramienta o con una metodología. La Figura A.1 muestra la planificación prevista.

Cronograma previsto

Fecha de inicio del proyecto: 27/02/2025
Incremento de desplazamiento: 3

Descripción	Inicio	Días
Planificación de Trabajo	27/02/2025	7
Búsqueda de literatura académica	06/03/2025	7
Búsqueda de literatura gris	13/03/2025	7
Elección de caso de estudio	20/03/2025	7
Caso de estudio	27/03/2025	14
Introducción	10/04/2025	7
Marco teórico	17/04/2025	14
Estado del Arte	01/05/2025	14
Conclusiones	15/05/2025	7
Anexos	22/05/2025	14
Revisión y correcciones	05/06/2025	14

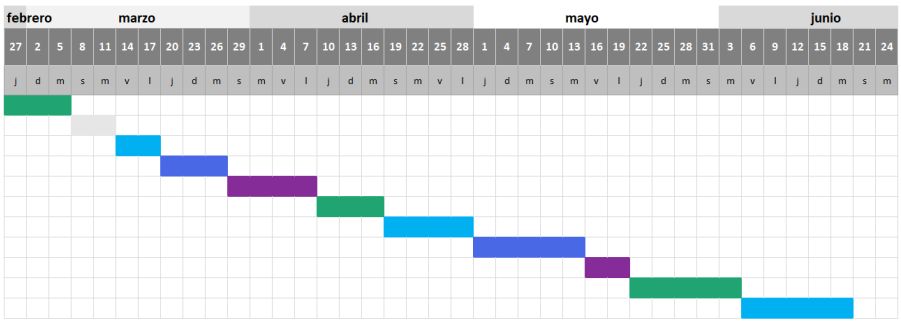

Figura A.1: Cronograma previsto al inicio del proyecto

A.2. Ejecución del trabajo

Durante la ejecución real se produjeron desviaciones respecto al plan. Se eligió una herramienta ya existente como caso práctico. No se preveía intervenir sobre ella, pero fue necesario adaptarla. Una parte del tiempo se destinó a la instalación, donde surgieron problemas con los drivers de GPU. El programa no usaba aceleración por hardware, por lo que se modificó para ejecutar modelos locales con soporte GPU. También se integraron nuevos modelos en la herramienta. **Se aplicaron medidas adicionales para cumplir con el reglamento europeo de protección de datos. Esto afectó la forma en que se gestionaban entradas, salidas y archivos temporales**. Estas tareas no estaban contempladas en la planificación original y ampliaron el tiempo total del proyecto. La Figura A.2 muestra el calendario ajustado con base en la ejecución real.

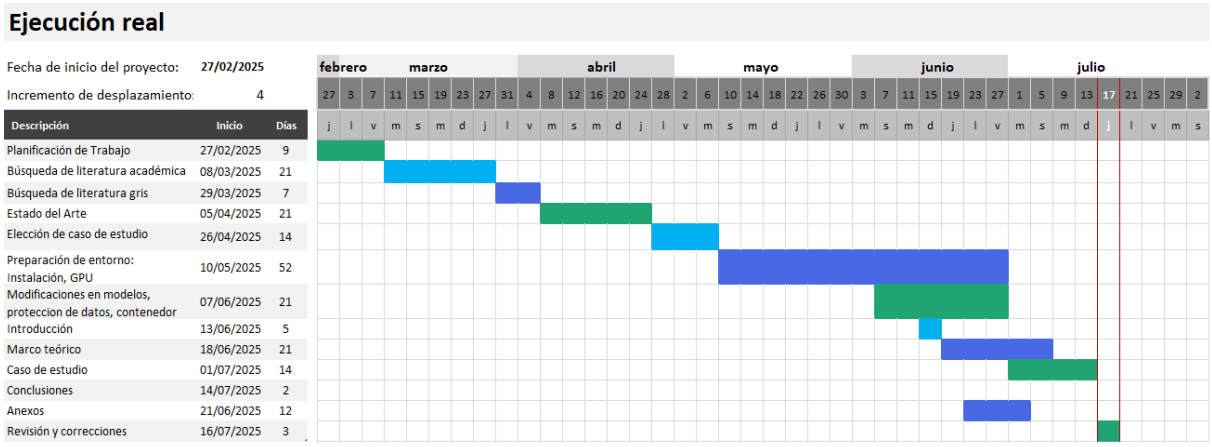

Figura A.2: Planificación real tras la ejecución del proyecto

Apéndice B

Especificación de Requisitos de Hardware

B.1. Introducción

Este apéndice recoge los requisitos de hardware necesarios para ejecutar los experimentos del TFM. Incluye las especificaciones del sistema base utilizado como entorno de pruebas.

B.2. Objetivos generales

Describir los recursos de hardware mínimos necesarios para disponer de un entorno reproducible, con capacidad de cómputo suficiente para ejecutar modelos generativos y tareas de procesamiento de datos a gran escala.

B.3. Catálogo de requisitos

En la tabla B.1 se detallan los componentes de la máquina virtual usada en los análisis y pruebas.

Componente	**Descripción**
Sistema operativo	Ubuntu 22.04
CPU	Intel(R) Xeon(R) Gold 5318Y @ 2.10GHz (2 cores, 2 sockets)
Memoria RAM	24.00 GiB
GPU	NVIDIA Quadro P4000 (8 GB VRAM)
Disco duro	100 GB

Tabla B.1: Especificaciones del sistema

Apéndice C

Guía técnica de instalación y uso

C.1. Introducción

Este apéndice documenta la instalación y uso de RetClean en un entorno local. El sistema se ejecuta mediante contenedores Docker que encapsulan los distintos componentes: modelo de lenguaje, backend, interfaz web y motor de búsqueda. Además, se habilita un entorno gráfico en la máquina virtual para acceder a la aplicación desde un navegador.

Los componentes que se van a instalar en esta sección se encuentran listados en la tabla C.1.

Paquete / Componente	**Descripción**
docker-ce	Motor de contenedores usado por el sistema.
docker-compose-plugin	Gestiona el arranque de contenedores definidos en archivos de configuración de docker compose.
build-essential	Requisito previo para instalar nvidia-container-toolkit.
nvidia-container-toolkit	Soporte para usar GPU dentro de Docker.
nvidia-driver-550	Driver para GPU A2.
xfce4	Entorno de escritorio ligero y rápido, usado para ejecutar la interfaz gráfica.
links2	Navegador usado para mostrar la interfaz web de la aplicación.
git	Herramienta de control de versiones usada para clonar el repositorio del proyecto.
retclean-analisis-tfm	Repositorio principal del sistema.

Tabla C.1: Paquetes y herramientas a instalar

C.2. Requisitos de usuarios

- Tener los requisitos de hardware que se detallan en el apéndice B.
- Disponer de los drivers adecuados de la GPU a utilizar. En caso de no disponer de estos drivers, contar con el ejecutable del driver de NVIDIA (nvidia-driver-550) antes de iniciar la instalación. En este caso fue proporcionado por el Departamento de Informática de la Universidad de Valladolid.

C.3. Instalación del sistema

Todos los componentes que se van a instalar en esta sección fueron obtenidos mediante APT desde repositorios oficiales, salvo el driver de GPU, que fue ejecutado manualmente. A continuación se describen los pasos realizados para la instalación de cada uno.

Actualizar la información del repositorio

```
sudo apt-get update
```

Configurar el repositorio de Docker

Se procede con la instalación de Docker[17]:

1. **Agregar la clave GPG de Docker**

   Se añade la clave oficial de Docker para permitir la verificación de los paquetes descargados:

   ```
   sudo apt-get install ca-certificates curl
   sudo install -m 0755 -d /etc/apt/keyrings
   sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /
       etc/apt/keyrings/docker.asc
   sudo chmod a+r /etc/apt/keyrings/docker.asc
   ```

2. **Añadir el repositorio de Docker**

   Este paso deja preparado el entorno para esta instalación y futuras actualizaciones.

   ```
   echo \
   "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings
       /docker.asc] https://download.docker.com/linux/ubuntu \
   $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}"
       ) stable" | \
   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
   sudo apt-get update
   ```

3. **Instalar la última versión de Docker y sus componentes**

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
    buildx-plugin docker-compose-plugin
```

4. **Reiniciar el servicio de Docker**

```
sudo service docker restart
```

5. **Verificar que Docker se instaló correctamente**

   Para comprobar que Docker funciona como se espera, se ejecuta una imagen de prueba.

```
sudo docker run hello-world
```

   Este comando descarga una imagen de test, la ejecuta en un contenedor y muestra un mensaje de confirmación. Se puede ver en la imagen C.1 un ejemplo de resultado exitoso.

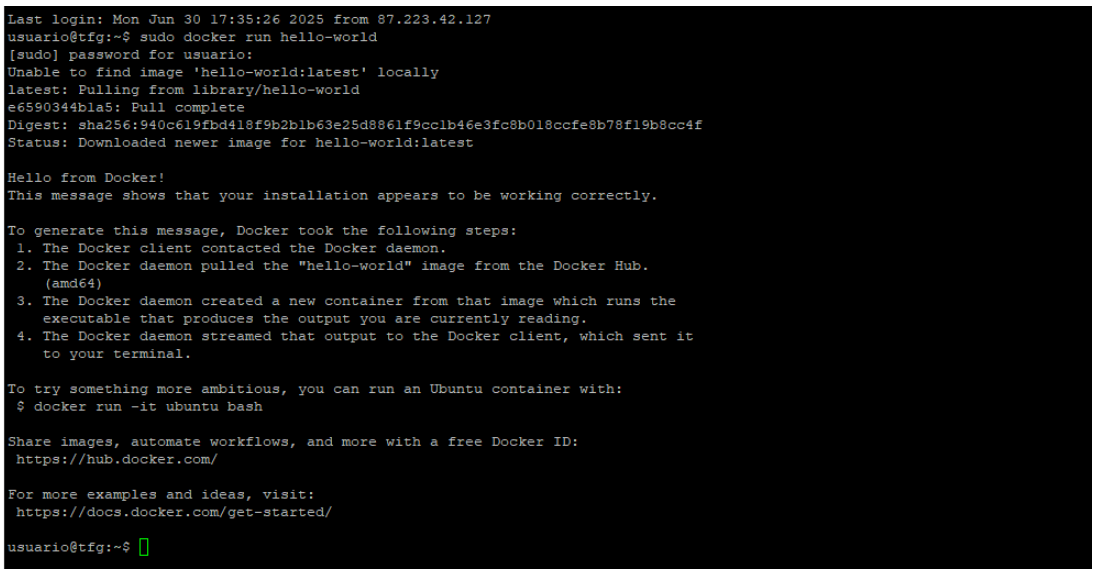

Figura C.1: Resultado del contenedor hello-world

Instalación del NVIDIA Container Toolkit

1. **Instalar herramientas de compilación**

   Se instala el paquete `build-essential`, requerido para compilar dependencias en caso de que no haya versiones precompiladas disponibles.

```
sudo apt-get install build-essential
```

2. **Instalar los drivers de la GPU**

   El controlador fue proporcionado por la Escuela de Ingeniería Informática y se encuentra en el directorio raíz del usuario root. Para instalarlo:

   ```
   sudo su -
   /root/NVIDIA-Linux-x86_64-550.54.14-grid.run
   ```

   Esto instala los drivers específicos para la GPU A2. Una vez instalado el driver de la GPU, se configura el repositorio de NVIDIA[40] desde donde se obtendrán los paquetes necesarios para habilitar el uso de la GPU dentro de contenedores Docker.

3. **Configurar el repositorio de NVIDIA**

   ```
   curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey |
      sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-
      toolkit-keyring.gpg \
     && curl -s -L https://nvidia.github.io/libnvidia-container/stable
        /deb/nvidia-container-toolkit.list | \
       sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-
          container-toolkit-keyring.gpg] https://#g' | \
       sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
   ```

4. **Actualizar la información de los repositorios**

   ```
   sudo apt-get update
   ```

5. **Instalar los paquetes del toolkit**

   ```
   export NVIDIA_CONTAINER_TOOLKIT_VERSION=1.17.8-1
     sudo apt-get install -y \
         nvidia-container-toolkit=${NVIDIA_CONTAINER_TOOLKIT_VERSION}
            \
         nvidia-container-toolkit-base=${
            NVIDIA_CONTAINER_TOOLKIT_VERSION} \
         libnvidia-container-tools=${NVIDIA_CONTAINER_TOOLKIT_VERSION}
             \
         libnvidia-container1=${NVIDIA_CONTAINER_TOOLKIT_VERSION}
   ```

6. **Configurar Docker para utilizar el driver de NVIDIA**

   Una vez instalado el toolkit, se configura Docker para que utilice el runtime de NVIDIA. Luego se reinicia el servicio para aplicar los cambios.

   ```
   nvidia-ctk runtime configure --runtime=docker
   sudo systemctl restart docker
   ```

Instalación del entorno de escritorio

Para interactuar con RetClean se habilita un entorno de escritorio. El elegido para este proyecto es XFCE[51], ya que proporciona una interfaz gráfica completa y ligera, y además facilita el control remoto del entorno gráfico mediante herramientas como X2Go.

1. **Instalar entorno de escritorio**

```
sudo apt-get install xfce4
```

2. **Verificar instalación**

   Para comprobar si XFCE se ha instalado correctamente, se puede verificar la presencia del ejecutable:

```
which startxfce4
```

   Si está instalado correctamente, este comando mostrará la ruta al ejecutable.

Instalación del navegador

1. **Instalar el navegador**

   Una vez instalado XFCE, se requiere un navegador ligero para acceder a la interfaz web de RetClean. Para este propósito se utiliza Links2[53], que permite abrir páginas desde terminal con soporte básico para modo gráfico.

```
sudo apt-get install links2
```

2. **Verificar instalación**

   Para confirmar que el navegador se instaló correctamente[52], se ejecuta:

```
links2 -version
```

   Debe mostrarse la versión instalada del navegador Links2.

Descarga de la aplicación RetClean

1. **Instalar Git**

   Se instala la herramienta `git`, necesaria para clonar el repositorio del proyecto.

```
sudo apt-get install git
```

2. **Cambio al directorio principal del usuario:**

```
cd /home/usuario
```

3. **Clonación del repositorio modificado del proyecto RetClean utilizado para este TFM::**

```
git clone https://gitlab.inf.uva.es/micplua/retclean-analisis-tfm.
    git
```

Este comando crea una carpeta llamada *retclean-analisis-tfm* en el directorio actual.

> **Nota**
>
> El repositorio *retclean-analisis-tfm* parte del original https://github.com/qcri/RetClean y fue adaptado para incluir modelos personalizados y nuevos flujos de limpieza.

Construir e iniciar la aplicación RetClean

1. **Entrar al directorio del proyecto**

   Una vez clonado el repositorio, se accede a la carpeta del proyecto.

   ```
   cd retclean-analisis-tfm
   ```

2. **Construir la aplicación con Docker Compose**

   Se construyen los servicios y sus dependencias.

   ```
   sudo docker compose build
   ```

   Este comando instala los servicios definidos en el archivo `docker-compose.yml`, incluyendo las dependencias necesarias para el cliente y el servidor.

3. **Iniciar la aplicación**

   Se ejecuta el entorno definido en Docker Compose.

   ```
   sudo docker compose up -d
   ```

Verificar el estado del entorno

- **Verificar que los contenedores estén activos**

  Se lista el estado de los contenedores en ejecución.

  ```
  sudo docker container ls
  ```

- **Consultar los LLMs locales disponibles**

  Para ver los LLMs locales disponibles en el contenedor de Ollama se ejecuta:

  ```
  sudo docker exec b3c81cefe452 ollama list
  ```

  Donde `b3c81cefe452` es el identificador del contenedor. Este valor cambia en cada ejecución. Para obtener el identificador actual, se utiliza `docker container ls`.

- **Verificar la disponibilidad de la GPU**

  Para comprobar que la GPU está disponible en el host:

  ```
  nvidia-smi
  ```

  Para verificar su disponibilidad dentro del contenedor:

  ```
  sudo docker exec retclean-ollama-1 nvidia-smi
  ```

> Nota
>
> La aplicación queda disponible en la dirección local http://localhost:3000.

C.4. Uso de la herramienta RetClean

La interfaz de RetClean se divide en dos módulos principales: *Data Repair* y *Datalake Index*. El primero permite trabajar con columnas específicas en un archivo CSV usando modelos generativos. El segundo permite construir un índice a partir de múltiples archivos CSV para utilizar como referencia contextual durante la reparación.

Módulo Data Repair

Este módulo permite cargar un archivo para luego configurar su proceso de reparación basado en modelos LLM y recuperación de contexto desde un índice. Las opciones disponibles en este módulo son :

- **Archivo corrupto (Corrupt Data)**: permite cargar el archivo CSV con los datos que se desean reparar. El archivo debe tener cabecera en la primera fila. El botón *Browse* abre el explorador de archivos para seleccionar el archivo. Una vez cargado, la tabla aparece en la parte derecha de la pantalla.
- **Entity Description**: campo de texto opcional para describir el tipo de entidad presente en los datos. Esta descripción puede usarse para mejorar la comprensión contextual del modelo, pero no es obligatoria para ejecutar una reparación.
- **Target Column**: permite seleccionar una columna del archivo cargado que se desea reparar. Solo se puede seleccionar una columna por ejecución. Es obligatorio definirla antes de iniciar el proceso.
- **Repair Values**: define el tipo de valores que serán considerados como erróneos o incompletos. Hay tres opciones:
  - **Any**: se intentará reparar cualquier celda con valores inconsistentes o incorrectos, aunque no esté vacía.

  - **Null**: solo se repararán las celdas vacías o con valores nulos.
  - **Custom**: permite definir una lista personalizada de valores que deben ser tratados como errores. Esta opción activa un campo adicional para ingresarlos.

- **Pivot Columns**: permite seleccionar una o más columnas que se usarán como contexto para ayudar al modelo a inferir el valor correcto en la columna objetivo. Se recomienda seleccionar atributos relacionados con la identidad o características del registro. El campo se despliega como una lista seleccionable.

- **Reasoner**: define el modelo generativo que se utilizará para proponer las reparaciones. Por defecto aparece GPT-4, pero en esta investigación no se encuentra en uso, ya que solo están activos modelos locales como Deepseek-R1, LLaMA 3.1, Mistral y Qwen-3.

- **Search Index Name**: permite seleccionar el índice previamente creado en el módulo *Datalake Index*. El índice contiene ejemplos tabulares que el modelo puede consultar para encontrar registros similares durante la reparación. Este campo es obligatorio si se desea usar recuperación semántica.

- **Index Type**: permite seleccionar el tipo de recuperación que se aplicará al buscar ejemplos en el índice. Hay dos opciones:

  - **Semantic**: realiza búsqueda vectorial basada en el significado del contenido.
  - **Syntactic**: realiza búsqueda textual más literal basada en coincidencias de tokens.

- **Reranker Type**: permite seleccionar el modelo encargado de reordenar los resultados obtenidos por el motor de búsqueda. Hay dos opciones:

  - **ColBERT**: utiliza codificadores de tipo BERT distribuidos y eficientes.
  - **Cross Encoder**: realiza comparación directa entre pares de ejemplos y el query.

- **START**: botón que inicia el proceso de reparación con las configuraciones definidas. Una vez pulsado, el sistema consulta el índice, aplica el modelo y presenta los resultados.

- **RetClean Results**: columna generada al finalizar la inferencia. Muestra la transformación realizada por el modelo para cada celda detectada como reparable. El usuario puede revisar y decidir si acepta o rechaza cada sugerencia.

- **APPLY REPAIRS**: botón que aplica todas las reparaciones seleccionadas. El resultado se refleja en la tabla principal.

- **EXPORT**: botón que permite descargar el archivo corregido en formato CSV. Se conserva la estructura original y se actualiza únicamente la columna objetivo con los valores aprobados.

Módulo Datalake Index

El módulo Datalake Index permite crear, actualizar o eliminar índices de referencia a partir de archivos CSV. Estos índices son usados por el módulo Data Repair para realizar recuperación contextual durante el proceso de reparación. Los datos cargados son estructurados como una base de conocimiento auxiliar y permiten mejorar la precisión de los modelos generativos al ofrecer ejemplos similares. Sus opciones son :

Create

Permite crear un nuevo índice a partir de una carpeta de archivos CSV. El proceso se detalla a continuación:

1. Pulsar la pestaña *Datalake Index* y seleccionar la opción *CREATE* en la parte superior izquierda.
2. Cargar una carpeta que contenga archivos CSV. Todos los archivos deben tener estructura tabular y cabeceras consistentes.
3. Asignar un nombre único al índice. Debe estar escrito en minúsculas. Este nombre será visible en el campo *Search Index Name* del módulo *Data Repair*.
4. Iniciar el proceso de indexación. El sistema analiza el contenido, extrae información relevante y construye un índice que combina motores de búsqueda sintáctica y semántica.

Los índices se almacenan localmente y pueden reutilizarse mientras el contenedor siga activo y no transcurran más de 60 minutos sin ejecución. Después de ese tiempo, los data lakes y sus índices se eliminan automáticamente.

Update

Permite actualizar un índice existente. Es útil cuando se agregan nuevos archivos CSV a una carpeta ya indexada o se desea refrescar el contenido del índice.

1. Seleccionar la opción *UPDATE*.
2. Escoger el índice a actualizar.
3. Cargar los nuevos archivos o seleccionar nuevamente la carpeta completa.
4. Confirmar la actualización. El sistema reemplaza el índice anterior por una nueva versión.

Esta opción garantiza que los modelos trabajen siempre con datos actualizados como referencia.

Delete

Permite eliminar un índice existente. Es útil para liberar espacio o mantener solo los índices necesarios.

1. Seleccionar la opción *DELETE*.
2. Elegir el índice que se desea eliminar.
3. Confirmar la eliminación. Esta acción es irreversible.

Una vez eliminado, el índice desaparecerá de la lista de selección en el módulo *Data Repair*.

Flujo de uso

Para ejecutar una reparación en los datos con RetClean se siguen los siguientes pasos. Cada acción se configura de forma secuencial en la interfaz:

1. Subir el archivo CSV dañado desde el campo Corrupt Data
2. (Opcional)Ingresar una descripción en Entity Description
3. Seleccionar la columna objetivo en Target Column
4. Definir el tipo de valores a reparar en Repair Values
5. Elegir las columnas de contexto en Pivot Columns
6. Seleccionar el modelo a usar en Reasoner
7. (Opcional) Configurar recuperación desde índice si se incluye contexto externo:
   - Seleccionar un índice creado en el módulo Datalake Index
   - Definir el tipo de recuperación asociado al índice: semántica o sintáctica
   - Seleccionar el modelo de reranqueo
8. Pulsar START para iniciar la reparación
9. Revisar los resultados generados en RetClean Results
10. Confirmar o ajustar los valores propuestos
11. Aplicar las correcciones con APPLY REPAIRS
12. Exportar el archivo corregido con EXPORT

Apéndice D

Estructura del repositorio y organización de archivos

La estructura del proyecto se organiza sobre las carpetas base de RetClean: backend, ollama, assets y frontend. Estas definen la arquitectura funcional del sistema. El directorio backend contiene la lógica del servidor, las API REST y los módulos de procesamiento central.

La carpeta datasets incluye el conjunto de datos original de cálculos biliares, una versión modificada con errores introducidos de forma controlada y una versión adaptada para pruebas en entornos tipo data lake. La carpeta `clear_data` contiene un script que ejecuta la limpieza periódica de datos. La carpeta models se monta directamente en el contenedor de Ollama para mantener los modelos cargados localmente y reducir el tiempo de inicio.

La carpeta resultados agrupa los scripts de análisis y los resultados generados. Contiene un script que combina columnas originales, columnas con errores y las predicciones realizadas. También incluye un script de evaluación y un repositorio organizado con los resultados de cada modelo según la variable analizada: BMI, HFA y VFA.

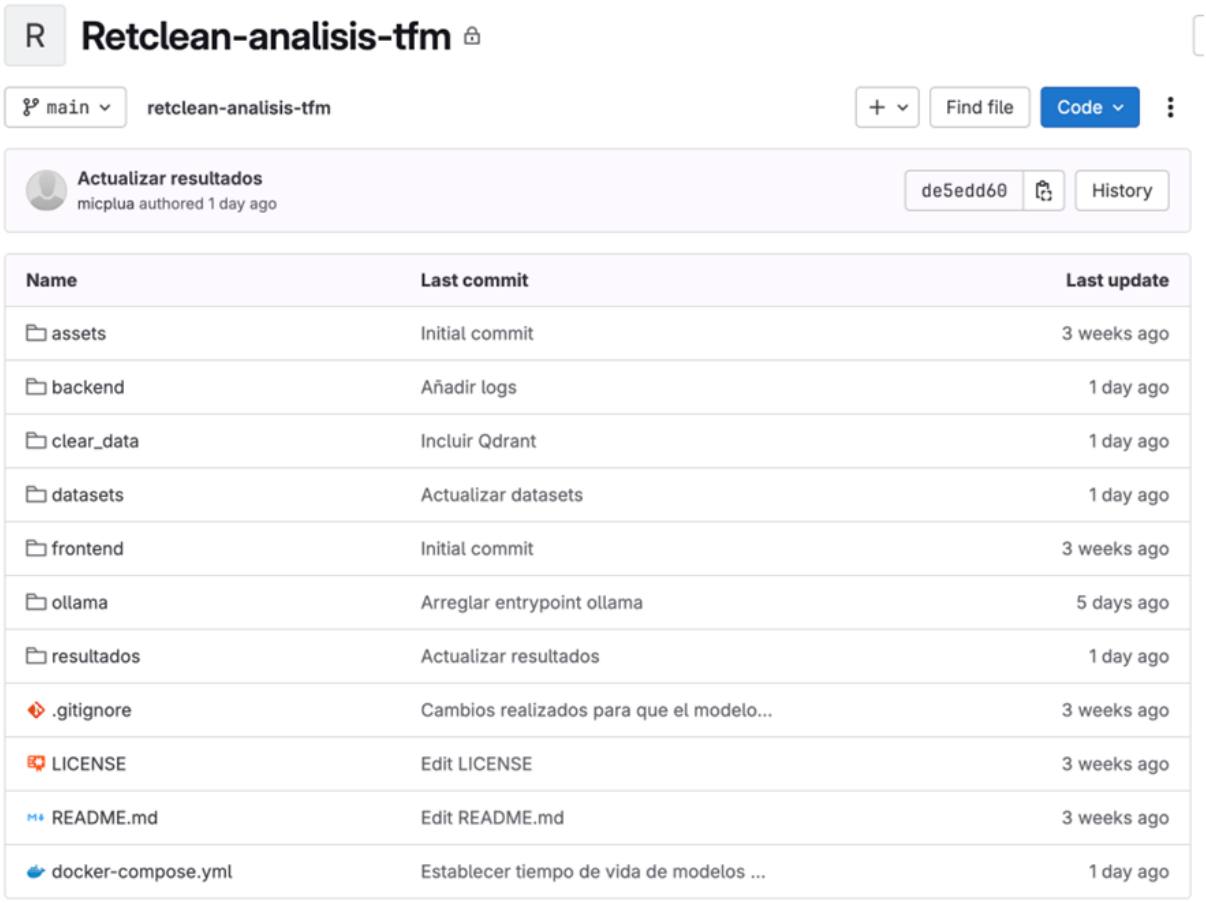

Figura D.1: Estructura de carpetas del repositorio adaptado