UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería de la Salud

Modelos de Lenguaje Pequeños para resumir Historias Clínicas: Comparativa, Destilación y RAG

Optimizing Small Language Models for Clinical Summarization: Comparison, Distillation, and RAG

Realizado por

Alejandro Silva Rodríguez

Tutorizado por

José Manuel Jerez Aragonés

Francisco Javier Moreno Barea

Departamento

Lenguajes y Ciencias de la Computación

MÁLAGA, junio de 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DE LA SALUD

Modelos de Lenguaje Pequeños para resumir Historias Clínicas: Comparativa, Destilación y RAG

Optimizing Small Language Models for Clinical Summarization: Comparison, Distillation, and RAG

Realizado por
Alejandro Silva Rodríguez
Tutorizado por
José Manuel Jerez Aragonés
Francisco Javier Moreno Barea

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2025

Fecha defensa: julio de 2025

Abstract

The automatic generation of clinical summaries from unstructured medical texts represents a significant challenge, as redundancy and disorganization in patient records can hinder effective decision-making. This study explores the use of Small Language Models (SLMs) as a cost-effective and accessible alternative to large-scale models, evaluating their performance in clinical summarization tasks. Advanced prompt engineering strategies were applied, with step-by-step generation proving particularly effective. Additionally, a multi-question Retrieval-Augmented Generation (RAG) approach was implemented to enhance contextual understanding. Fine-tuning of the LLaMA 3.2 model revealed effective learning but also stylistic overfitting, highlighting the need for a corpus better aligned with clinical writing. Automatic evaluation was complemented by expert validation, and a functional web application was developed to integrate the optimized model, enabling practical generation and assessment of clinical summaries. Overall, the study validates a reproducible workflow and demonstrates the real-world potential of SLMs in clinical environments, despite their limitations compared to larger models like GPT-4o-mini.

Keywords: **Clinical summarization, Natural Language Processing (NLP), Small Language Models**

Resumen

La generación automática de resúmenes clínicos a partir de textos no estructurados representa un desafío relevante en el ámbito médico, donde la redundancia y la desorganización de los historiales pueden dificultar la toma de decisiones. Este trabajo explora el uso de modelos de lenguaje pequeños (Small Language Models, SLMs) como alternativa eficiente y accesible a los modelos de gran escala, evaluando su rendimiento en tareas de síntesis clínica. Se han aplicado estrategias avanzadas de prompt engineering, siendo especialmente efectiva la generación por partes, y se ha incorporado un enfoque de Recuperador-Generador (RAG) con preguntas múltiples para enriquecer el contexto del modelo. Además, se ha realizado un ajuste fino sobre LLaMA 3.2, evidenciando buen aprendizaje pero con sobreajuste estilístico, lo que resalta la importancia de contar con un corpus clínicamente alineado. La evaluación automática se ha complementado con validación experta, y se ha desarrollado una aplicación web funcional que integra el modelo optimizado, permitiendo la generación y validación de resúmenes en un entorno práctico. En conjunto, el estudio valida un flujo de trabajo reproducible y muestra el potencial real de los SLMs en contextos clínicos, a pesar de sus limitaciones frente a modelos como GPT-4o-mini.

Palabras clave: Resumen clínico automático, Procesamiento del lenguaje natural (PLN), Modelos de lenguaje pequeños

Índice

1

Introducción

1.1. Motivación

Las consultas clínicas e historiales reales suelen presentarse en forma de textos no estructurados, donde los profesionales de la salud frecuentemente copian y pegan información de consultas previas. Este método genera redundancia en los datos y dificulta la identificación de la información clínica actualizada y relevante, lo que puede impactar negativamente en la toma de decisiones médicas y en la eficiencia de los sistemas de gestión hospitalaria (Searle et al., 2021).

En los últimos años, los modelos de lenguaje de gran escala (LLMs, por sus siglas en inglés) han demostrado avances notables en la generación de resúmenes de alta calidad a partir de textos extensos (Y. Zhang et al., 2025). Sin embargo, estos modelos suelen ser de gran tamaño y, en muchos casos, su uso está limitado por costos asociados a licencias propietarias. Como respuesta a esta limitación, se ha promovido el desarrollo de modelos de lenguaje pequeños (Small Language Models, SLMs), los cuales ofrecen accesibilidad y menores requerimientos computacionales. No obstante, estos modelos presentan desafíos significativos, como la omisión de información clave y entidades relevantes, especialmente cuando los documentos de entrada son extensos (Grail et al., 2021; G. Zhang et al., 2024).

Este trabajo surge de la necesidad de explorar soluciones eficientes para la generación de resúmenes clínicos que mantengan la relevancia de la información y sean accesibles en términos de costo y recursos computacionales. El dominio de aplicación de este estudio se encuentra en el procesamiento del lenguaje natural (NLP) aplicado al ámbito médico, con un enfoque en la mejora de los resúmenes clínicos generados por modelos de lenguaje pequeños.

1.2. Marco de Investigación

Aunque este trabajo no se desarrolla directamente dentro del grupo de investigación de inteligencia computacional en biomedicina de la universidad de Málaga, guarda una estrecha relación con una de sus líneas actuales y pretende colaborar en el avance de la misma. El grupo Inteligencia Computacional en Biomedicina, liderado por el Dr. José Manuel Jerez Aragonés, pertenece al Departamento de Lenguajes y Ciencias de la Computación de la E.T.S.I. de Informática y está adscrito al Instituto Universitario de Investigación en Tecnologías Lingüísticas Multilingües (IUITLM).

Desde su fundación en enero de 2012, el grupo se ha centrado en el diseño de algoritmos de Inteligencia Artificial y Procesamiento del Lenguaje Natural para el análisis automático de información no estructurada, especialmente la contenida en historias clínicas electrónicas.

1.3. Objetivos

Objetivo General

Desarrollar y evaluar modelos de lenguaje pequeños (Small Language Models, SLMs) en la generación de resúmenes clínicos, comparándolos con modelos de mayor escala como GPT-4, y optimizando su desempeño mediante técnicas avanzadas de procesamiento del lenguaje natural.

Objetivos Específicos

1. **Evaluación del desempeño de SLMs**
    - Analizar la capacidad de SLMs en la generación de resúmenes clínicos.
    - Comparar su rendimiento con modelos de mayor escala como GPT-4.
2. **Optimización de la generación de resúmenes**
    - Identificar técnicas que permitan mejorar la calidad de los resúmenes generados por SLMs.
    - Implementar estrategias de *prompt engineering* para optimizar la generación de resúmenes.

- Aplicar técnicas avanzadas como RAG (Recuperador Generador), *fine-tuning* y destilación de conocimiento.

3. **Validación y evaluación de resultados**

   - Evaluar la calidad y relevancia de los resúmenes mediante la consulta con expertos médicos.
   - Aplicar métricas especializadas para medir la precisión y coherencia de los resúmenes:
     - Métricas de superposición: ROUGE, BLEU, METEOR.
     - Métricas de similitud semántica: BERTScore.
     - Evaluaciones basadas en SLMs para un análisis más preciso.

4. **Desarrollo de una aplicación prototipo**

   - Diseñar y desarrollar una aplicación web que implemente el mejor modelo identificado para la generación automática de resúmenes de historiales clínicos.

1.4. Estructura del documento

1. **Antecedentes**: Un repaso de trabajos anteriores y aclaración de terminología.

2. **Generación y validación de resúmenes modelo**: Generación de resúmenes de referencia utilizando GPT-4 y validación con un profesional médico para establecer un punto de comparación.

3. **Desarrollo y optimización de modelos SLMs**: Uso de modelos de lenguaje pequeños (SLMs) y técnicas de *prompt engineering* para la síntesis de historiales clínicos.

4. **Evaluación comparativa**: Comparación de los resúmenes generados por SLMs con los de GPT-4. Para ello, se emplearán métricas especializadas como ROUGE, BLEU y BERTScore, asegurando una evaluación objetiva de la calidad del texto.

5. **Implementación de modelo final**: Aplicación de destilación del conocimiento de modelo GPT-4 hacia el mejor Slm e implementación de RAG.

6. **Comparación final y validación con expertos**: Realización de evaluación de los resultados obtenidos con respecto a los resúmenes de referencia, además de contrastarlos con la opinión de profesionales médicos para garantizar la utilidad y precisión del sistema.

7. **Desarrollo del prototipo de aplicación web**: Diseño e implementación de una aplicación web en la que se integra el mejor modelo seleccionado para la generación automática de resúmenes clínicos, permitiendo su validación en un entorno práctico.

1.5. Tecnologías usadas

En la tabla 1 se encuentran las tecnologías y recursos usados en la elaboración de este trabajo.

Cuadro 1: Lista de tecnologías utilizadas en el proyecto.

Tecnologías usadas
Python
PyTorch
Hugging Face
Ollama
Unsloth
ChromaDB
FastAPI
JavaScript
Visual Studio Code
Tarjeta de video Nvidia RTX 3050 Laptop

2

Antecedentes

2.1. Aproximación al Problema

La evolución de los historiales médicos comenzó en la antigüedad con la creación de informes escritos sobre casos para fines didácticos. En el siglo XIX, en ciudades como París y Berlín, surgieron los primeros antecedentes de los registros médicos modernos, y en Estados Unidos, los hospitales de enseñanza impulsaron su desarrollo. Sin embargo, no fue hasta el siglo XX cuando se estableció un historial clínico útil para el cuidado directo del paciente en hospitales y ambulatorios (Gillum, 2013).

La historia clínica electrónica (HCE) es un sistema diseñado para la digitalización de documentos médicos, incluyendo resultados de pruebas, prescripciones e imágenes. Su implementación tiene como objetivo mejorar la atención sanitaria, reducir costos y minimizar fraudes. Sin embargo, su adopción ha generado resistencias, principalmente debido a la carga documental adicional que representa para los profesionales de la salud.

Además de su impacto en la gestión clínica, la HCE desempeña un papel clave en la salud pública, facilitando la integración de información ambiental y la estandarización de protocolos médicos. No obstante, la masiva generación de datos derivados de estos sistemas plantea un desafío significativo en términos de almacenamiento, procesamiento y análisis (Cyganek et al., 2016).

En este contexto, el procesamiento de lenguaje natural (PLN) se ha consolidado como una técnica fundamental en la informática clínica, especialmente para extraer y estructurar datos no estructurados provenientes de los HCE, como los resúmenes de alta. El PLN se refiere a un subcampo de la inteligencia artificial que permite a las máquinas comprender, interpretar y generar lenguaje humano de manera que sea útil (Chopra et al., 2013). Esta capacidad de estructuración es crucial en estudios de uso secundario de datos EHR, permitiendo la conversión

de información narrativa en datos computables.

Recientemente, el PLN ha sido aplicado con éxito en la extracción de información y en la identificación de eventos adversos por medicamentos (ADE) en los datos de HCE. La extracción de información (IE, por sus siglas en inglés) es una de las aplicaciones más tradicionales del PLN, orientada a identificar y clasificar entidades nominales, como conceptos y afirmaciones, así como sus relaciones dentro de textos narrativos (Frey et al., 2014).

También se ha avanzado en otras técnicas de procesamiento de HCE como la desidentificación (Moreno-Barea et al., 2025).

2.2. Trabajos Relacionados

En esta sección se abordará la evolución del uso de PLN en la síntesis de historia clínica.

Inicialmente, los modelos LLM demostraron su capacidad para almacenar y manipular conocimiento factual. Sin embargo, su rendimiento en tareas intensivas en conocimiento era limitado debido a su incapacidad para acceder y actualizar información específica de manera eficiente. Para abordar esta limitación, se propuso la técnica de RAG, que combina modelos de lenguaje preentrenados con mecanismos de recuperación de información no paramétrica, mejorando así la generación de texto en tareas que requieren conocimiento específico (Lewis et al., 2021).

En el contexto clínico, esta aproximación ha sido explorada entre otros por Saba et al., quienes propusieron un sistema que combina recuperación semántica, RAG y generación de respuestas a preguntas específicas definidas por expertos médicos. Esta metodología busca generar resúmenes centrados en preguntas clave, evitando las limitaciones de atención que presentan los LLMs con entradas largas y mitigando problemas comunes como las alucinaciones. El enfoque permite generar información diversa y precisa sin requerir entrenamiento adicional, lo que lo convierte en una alternativa eficaz para la síntesis de HCE (Saba et al., 2024).

Paralelamente, el ajuste fino de LLM en datos clínicos específicos ha sido fundamental para mejorar su rendimiento en tareas médicas. Tinn et al. realizaron un estudio sistemático sobre la estabilidad del ajuste fino en PLN biomédico, identificando técnicas que mejoran significativamente el rendimiento en aplicaciones con pocos recursos (Tinn et al., 2023).

Un ejemplo destacado de ajuste fino en el ámbito clínico es el trabajo de Guluzade et al.,

quienes presentaron ELMTEX, un modelo ajustado para extraer información estructurada de informes clínicos. Su estudio demostró que modelos más pequeños y ajustados pueden igualar o superar a modelos más grandes en entornos con recursos limitados (Guluzade et al., 2025).

Además, Davis et al. desarrollaron MedSlice, un modelo ajustado para la segmentación segura de notas clínicas, que superó a modelos propietarios como GPT-4o y GPT-4o mini en precisión y accesibilidad. En su estudio, se enfocaron en extraer secciones clave de las notas clínicas, como el historial de la enfermedad actual, el historial intermedio y la evaluación y plan. Utilizando un conjunto de datos de 487 notas de progreso, compararon el rendimiento de modelos de lenguaje de código abierto ajustados, como Llama 3.1 8B, con los modelos propietarios mencionados. Los resultados mostraron que el modelo ajustado Llama 3.1 8B alcanzó una puntuación F1 de 0.92, superando a los modelos propietarios en la tarea de segmentación de notas clínicas (Davis et al., 2025).

Otros enfoques recientes han explorado variantes más especializadas. Zhang et al. propusieron NBCE, un mecanismo de extensión dinámica de contexto para mejorar la calidad de los resúmenes de historias clínicas extensas, alcanzando un desempeño cercano al modelo Gemini de Google (175B) en métricas ROUGE-L con un uso significativamente menor de recursos computacionales (G. Zhang et al., 2024).

Por su parte, Ryu et al. introdujeron KEITSum, un método de ajuste para sLLMs que incorpora instrucciones informadas por elementos clave del documento. Esta técnica mejora la relevancia del contenido resumido, reduciendo omisiones y alucinaciones, y acercando su desempeño al de modelos de gran escala (Ryu et al., 2024).

Adicionalmente, se ha explorado la distilación de conocimiento como estrategia para trasladar capacidades de LLMs complejos hacia modelos más pequeños y eficientes. Zhang et al. revisan extensamente este enfoque en el contexto clínico, destacando su utilidad para mantener precisión sin los elevados costos computacionales de los modelos originales (Y. Zhang et al., 2025).

Finalmente, técnicas basadas en redes pointer-generator también han sido aplicadas a la generación automática de resúmenes de altas médicas. Estas redes permiten copiar directamente términos del texto original mientras generan nuevo contenido, lo que mejora la fidelidad semántica de los resúmenes generados (B. Zhao et al., 2024).

2.3. Tecnología

Los Modelos de Lenguaje de Gran Escala (LLM) son modelos de inteligencia artificial diseñados para trabajar con grandes volúmenes de texto y generar o comprender lenguaje humano. Estos modelos se entrenan utilizando enormes cantidades de datos textuales y tienen una arquitectura capaz de capturar relaciones complejas y contextos a largo plazo entre las palabras (W. X. Zhao et al., 2023).

2.3.1. Arquitectura de los Modelos de Lenguaje de Gran Escala

- **Transformers**: La base de los LLMs modernos es la arquitectura Transformer, propuesta por Vaswani (Vaswani et al., 2017). Este modelo ha demostrado ser muy efectivo para tareas de procesamiento de lenguaje natural, gracias a su capacidad para capturar relaciones contextuales entre las palabras en una secuencia sin depender de la secuencialidad de los datos. En lugar de procesar palabras de manera secuencial como en las redes neuronales recurrentes (RNN), el Transformer considera la totalidad de la secuencia de entrada simultáneamente, lo que permite que el modelo capture dependencias a largo plazo y relacione todas las palabras de manera eficiente, pudiendose paralelizar el proceso de entrenamiento e inferencia.

- **Mecanismo de Auto-Atención y Multi-head Attention**: El componente clave en un Transformer es el mecanismo de auto-atención, que permite al modelo evaluar la importancia de cada palabra en relación con todas las demás palabras dentro de una secuencia.

  Para lograr esto, cada palabra en la secuencia se transforma en tres representaciones diferentes: **Q** (Query), **K** (Key) y **V** (Value). La matriz de **Q** representa la palabra que realiza la consulta, la matriz de **K** contiene las palabras a comparar y la matriz de **V** contiene los valores que se combinan según la importancia determinada. La puntuación de atención se obtiene mediante el producto escalar entre **Q** y **K**, que luego se normaliza con una función softmax para asignar pesos a **V**, generando así la representación contextual de cada palabra.

  El modelo de **multi-head attention** permite que el Transformer procese diferentes

aspectos del contexto de manera simultánea, representando diversas interpretaciones de las relaciones entre las palabras. Cada cabeza de atención se enfoca en diferentes partes del texto, lo que enriquece la comprensión del modelo sobre los matices semánticos y sintácticos de las palabras (Chopra et al., 2013).

- **Codificación y Decodificación**: En un Transformer, el proceso se divide en dos fases: codificación y decodificación. El codificador toma la secuencia de entrada, la convierte en una representación interna (vectorial), y el decodificador utiliza esta representación para generar la salida deseada (Vaswani et al., 2017).

2.3.2. Pre-entrenamiento, Fine-Tuning y Destilación de Conocimiento

El pre-entrenamiento permite a los LLMs aprender patrones lingüísticos a partir de grandes corpus de texto no etiquetados. Técnicas como Masked Language Modeling (MLM) en BERT y Causal Language Modeling (CLM) en GPT ayudan a capturar la estructura del lenguaje. Posteriormente, el ajuste fino adapta el modelo a tareas específicas mediante conjuntos de datos más pequeños y etiquetados, optimizando su desempeño en aplicaciones concretas (Ren & Sutherland, 2024).

La destilación de conocimiento es un enfoque para transferir conocimiento de un modelo grande (profesor) a uno más pequeño (estudiante), manteniendo un rendimiento competitivo con menor costo computacional (Sreenivas et al., 2025).

2.3.3. Generación Aumentada por Recuperación

El enfoque de Generación Aumentada por Recuperación, conocido en inglés como Retrieval-Augmented Generation (RAG) combina la generación de texto con la recuperación de información externa, mejorando la precisión y relevancia de las respuestas. Para ello, el modelo busca información en bases de datos externas y la usa como contexto antes de generar una respuesta.

Un elemento clave en RAG es el uso de **bases de datos de vectores densas**, donde documentos o fragmentos de información se almacenan como representaciones numéricas de alta dimensión. La recuperación eficiente de información en estos espacios se logra mediante **técnicas de partición**, como product quantization o hierarchical clustering, que aceleran la

búsqueda sin comprometer la calidad (Lewis et al., 2020).

3

Metodología

3.1. Generación y validación de resúmenes modelo

Con el objetivo de evaluar la calidad de los resúmenes generados automáticamente, se elaboró un conjunto de resúmenes modelo que sirvieron como referencia en el proceso de validación.

Estos resúmenes se generaron a partir de historiales médicos anonimizados combinados con una instrucción común, que especificaba el estilo y el nivel de detalle esperado. Tanto los historiales como la instrucción fueron preparados previamente en archivos de texto plano para facilitar su procesamiento.

Para la generación automática de los resúmenes se empleó la API de OpenAI, utilizando el modelo gpt-4o-mini en modo de procesamiento por lotes (batch). Este modo permite enviar múltiples solicitudes en un solo envío, lo que resulta especialmente útil cuando se trabaja con grandes volúmenes de datos. Cada solicitud incluía un historial clínico y el prompt correspondiente (OpenAI, 2024).

Una vez enviado el lote, se monitorizó su estado hasta que todas las respuestas estuvieron disponibles, las cuales fueron posteriormente organizadas y vinculadas con sus respectivos historiales originales.

Finalmente, una muestra de los resúmenes generados fue revisada por profesionales del Hospital Clínico Universitario de Málaga, con el objetivo de que confirmasen que los textos resultantes contenían la información relevante de forma clara y adecuada, validando así su uso como referencia en la evaluación del sistema.

3.2. Comparación de modelos SLM con resúmenes de referencia

Tras generar un conjunto de resúmenes modelo utilizando la API de OpenAI, se evaluó el rendimiento de distintos modelos de lenguaje de menor tamaño con el objetivo de determinar qué modelos eran capaces de aproximarse con mayor fidelidad a los resúmenes de referencia generados por GPT-4o-mini. Además, se midió el tiempo de inferencia de cada modelo.

3.2.1. Modelos utilizados

Los modelos de la tabla 2 se ejecutaron localmente mediante la herramienta ollama, que permite la interacción con modelos preentrenados de forma eficiente y a alto nivel. Ya que el objetivo de este trabajo es utilizar la cantidad mínima de cómputo, la ventaja más notable de esta herramienta es que implementa automáticamente el particionado del modelo entre GPU y CPU (conocido en inglés como offloading) si es necesario.

Cuadro 2: Lista de modelos utilizados en el proyecto.

Modelo	Parámetros	Tamaño de ventana	Cuantización
qwen:14b	14b	32768 tokens	Q4_0
llama3.1	8b	131072 tokens	Q4_K_M
llama3.2	3b	131072 tokens	Q4_K_M
deepseek-r1:14b	14b	131072 tokens	Q4_K_M
deepseek-r1:8b	8b	131072 tokens	Q4_K_M
mistral	7b	32768 tokens	Q4_0
phi4	14b	16384 tokens	Q4_K_M

Para cada historial clínico, se utilizaron diferentes contextos o instrucciones del sistema, almacenadas en archivos de texto, que se incorporaron como mensajes del sistema al inicializar el modelo. Se evaluaron dos configuraciones distintas de temperatura (0.2 y 0.8) para estudiar su efecto en la generación de texto.

3.2.2. Técnicas de Prompting utilizadas

Durante las pruebas realizadas con distintos modelos de lenguaje, se aplicaron diversas metodologías de prompting con el objetivo de optimizar la calidad de las respuesta. A continuación se detallan las principales estrategias empleadas, junto con su justificación:

- **Template-based prompting**: Se diseñaró una plantilla con la estructura que debe tener el resumen. Esta técnica, al ser consistente en contenido fue usada en la obtención de resúmenes modelos en el paso anterior. Esta técnica permite estandarizar el formato de entrada y aprovechar estructuras de lenguaje que los modelos ya han visto durante su entrenamiento (Brown et al., 2020).

- **Chain-of-Thought (CoT)**: Se utilizó esta estrategia para inducir razonamiento complejo. De esta forma se mejora la capacidad para resolver problemas en etapas, especialmente en tareas lógico-matemáticas o de toma de decisiones secuencial (Wei et al., 2023).

- **Few-shot prompting**: Se introdujo un ejemplo dentro del prompt para mejorar la generalización del modelo en tareas específicas. Los modelos de lenguaje suelen tener buenos resultados si tienen ejemplos a seguir (Parnami & Lee, 2022).

- **Stepwise refinement**: Esta técnica consistió en dividir tareas complejas en subtareas más pequeñas que eran resueltas en etapas consecutivas. Se aplicó principalmente cuando el modelo cometía errores por falta de planificación en tareas extensas. Su efectividad ha sido destacada en investigaciones sobre razonamiento iterativo (Sun et al., 2024).

- **Generación por partes**: Dado que algunos modelos tienen ventanas de contexto limitadas (por ejemplo, 4k tokens), se optó por dividir la entrada y salida en bloques lógicos. Hacer resúmenes de partes del historial y juntarlos en uno resulta costoso pero puede hacer que se mantenga coherencia en textos largos (Press et al., 2023).

3.2.3. Evaluación automática

Las respuestas generadas fueron comparadas con las respuestas de referencia mediante dos tipos de métricas automáticas comúnmente utilizadas en tareas de procesamiento de lenguaje natural:

- **BLEU** (Bilingual Evaluation Understudy): mide la superposición de n-gramas entre la respuesta generada y la esperada. Se utilizó la variante con suavizado de Chencherry (Papineni et al., 2002).

- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation): se calcularon las variantes ROUGE-1, ROUGE-2 y ROUGE-L, que capturan la coincidencia de unigramas, bigramas y la subsecuencia más larga común respectivamente (Lin, 2004).

- **BERTScore**: mide la similitud semántica entre la respuesta generada y la referencia utilizando los embeddings contextuales de BERT. Calcula la similitud coseno entre los vectores de las palabras en ambos textos, y se reportan las métricas de precisión, recuerdo y F1 (T. Zhang et al., 2020).

Estas fueron implementadas usando las bibliotecas nltk y rouge-score de Python.

3.3. Implementación del modelo final

3.3.1. RAG

Una vez seleccionado el mejor modelo, se buscó optimizar aún más su rendimiento mediante la implementación de un sistema de Retrieval-Augmented Generation (RAG).

Este enfoque consistió en proporcionar contexto adicional al modelo a partir de una base de datos de preguntas y respuestas médicas, con el objetivo de mejorar la precisión en la tarea de resumir historias clínicas. Aunque este método se emplea comúnmente en aplicaciones de asistencia conversacional, ha demostrado ser eficaz también en la generación de resúmenes clínicos (Alkhalaf et al., 2024).

Se utilizó un conjunto de datos compuesto por 194,000 preguntas y respuestas del dataset MEDMCQA (Pal et al., 2022), las cuales fueron transformadas en vectores de embeddings y almacenadas en una base de datos vectorial utilizando ChromaDB.

La Figura 1 presenta el diagrama de flujo del sistema implementado, que sigue los siguientes pasos:

1. El historial clínico se divide en secciones, siguiendo el mismo esquema empleado en la generación por partes, para facilitar una búsqueda ordenada de información.

2. Se solicita al modelo que identifique las ideas principales de cada sección del texto, con el fin de usarlas como claves de búsqueda.

3. Cada una de estas ideas se consulta en la base de datos vectorial, obteniendo definiciones o respuestas a preguntas relacionadas.

4. A continuación, se resume cada sección del historial clínico, incorporando como contexto la información obtenida en el paso anterior.

5. Finalmente, los resúmenes parciales se fusionan utilizando el modelo para generar un resumen consolidado de la historia clínica.

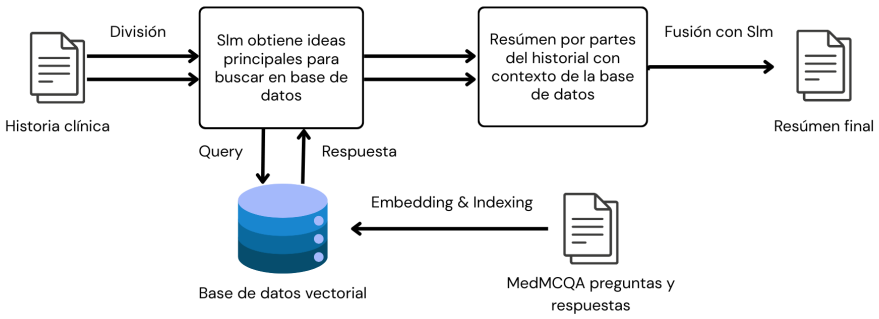

Figura 1: Diagrama de flujo del sistema RAG.

3.3.2. Ajuste fino del modelo

Debido a las limitaciones en los recursos computacionales disponibles, se optó por ajustar el modelo más pequeño entre los evaluados, específicamente el llama3.2:3b-instruct, aplicando una cuantización de 4 bits para reducir el uso de memoria y facilitar el entrenamiento en hardware limitado.

La cuantización es una técnica que reduce la precisión de los parámetros del modelo, disminuyendo así su tamaño y los requisitos de memoria. En particular, la cuantización a 4 bits permite representar los pesos del modelo utilizando solo 4 bits por valor, en lugar de los 32 bits

típicos en punto flotante. Esto resulta en una compresión significativa del modelo, permitiendo su ejecución en dispositivos con recursos limitados (Face, 2023).

Para ajustar el modelo cuantizado, se empleó la técnica de Low-Rank Adaptation (LoRA), que introduce adaptadores de bajo rango en las capas del modelo. Esta metodología permite modificar solo una pequeña fracción de los parámetros, manteniendo el resto del modelo congelado, lo que reduce considerablemente los requisitos computacionales y de memoria durante el entrenamiento (Hu et al., 2021).

El proceso de ajuste fino se llevó a cabo utilizando la biblioteca Unsloth, una herramienta de código abierto diseñada para facilitar y optimizar el entrenamiento de modelos de lenguaje grandes en entornos con recursos limitados. Unsloth proporciona implementaciones eficientes de técnicas como LoRA y cuantización, y maneja automáticamente la gestión de memoria y el procesamiento de datos durante el entrenamiento (Team, 2025).

Uno de los desafíos más significativos fue comprender y aplicar correctamente las plantillas de prompts. Estas plantillas definen la estructura de las entradas que se proporcionan al modelo durante el entrenamiento y la inferencia, y son cruciales para guiar el comportamiento del modelo de manera coherente. La correcta implementación de estas plantillas asegura que el modelo interprete y responda adecuadamente a las instrucciones proporcionadas (Face, 2025).

El entrenamiento se realizó utilizando Unsloth con los siguientes parámetros:

- Número de ejemplos: 256,916
- Épocas: 1
- Tamaño de batch por dispositivo: 2
- Pasos de acumulación de gradiente: 4
- Tamaño total de batch: 8
- Parámetros entrenables: 24,313,856 de un total de 3,000,000,000 (0.81 % del modelo)
- Tiempo total de entrenamiento: aproximadamente 6 minutos

Durante el entrenamiento, Unsloth gestionó de manera eficiente la memoria, descargando gradientes según fuera necesario para optimizar el uso de VRAM.

Los datos utilizados para el ajuste fino provienen del repositorio ruslanmv/ai-medical-chatbot (Vsevolodovna, 2024), que contiene un conjunto de datos especializado en conversaciones médicas. Este conjunto de datos proporcionó ejemplos relevantes para adaptar el modelo a tareas específicas en el ámbito de la medicina.

3.4. Desarrollo de la aplicación web demostrativa

Como parte de la validación práctica del sistema propuesto, se ha desarrollado una aplicación web simple que permite interactuar con el modelo de lenguaje entrenado. Esta aplicación permite introducir un historial clínico y obtener como respuesta un resumen generado automáticamente.

3.4.1. Tecnologías utilizadas

- **FastAPI**: framework de backend utilizado para definir la API REST que comunica al usuario con el modelo.
- **Ollama**: herramienta para ejecutar modelos de lenguaje localmente de forma eficiente.
- **HTML, CSS y JavaScript**: usados para construir la interfaz web de forma sencilla y funcional.

3.4.2. Endpoints de la API

La API cuenta con los siguientes endpoints:

- POST /predict: recibe un historial médico en formato texto plano y devuelve el resumen generado por el modelo.
- GET /: endpoint que sirve los archivos estáticos (HTML, CSS y JavaScript) de la web.

3.4.3. Funcionamiento general

Cuando el usuario envía un historial médico desde el formulario web, este se envía al endpoint /predict. El backend lo procesa y consulta al modelo cargado en Ollama, que genera la respuesta. Esta es devuelta al usuario en la misma página web.

4

Resultados y Discusión

4.1. Generación y validación de resúmenes modelo

Se generaron un total de 50 resúmenes a partir de los historiales clínicos seleccionados. Los expertos evaluaron la calidad de los resúmenes en términos de precisión, claridad y completitud de la información. Aunque eran distinguibles a simple vista con los resúmenes hechos a manos por los profesionales, estos contennían los contenidos y estructura adecuada.

Estos resultados respaldan el uso de los resúmenes modelo como referencia válida para la posterior evaluación automática del sistema de generación.

4.2. Comparación de modelos SLM con resúmenes de referencia

En esta sección se presentan los resultados obtenidos al evaluar los distintos modelos de lenguaje de menor tamaño en comparación con los resúmenes de referencia generados por GPT-4o-mini. El rendimiento de cada modelo se midió utilizando varias métricas automáticas de similitud, así como la complejidad temporal asociada con cada uno de los modelos.

4.2.1. Métricas de similitud

Se obtuvieron las métricas de similitud entre las 3000 respuestas generadas y las respuestas de referencia. Las métricas utilizadas incluyen BLEU, ROUGE y BERTScore.

Los resultados de las métricas de similitud por temperatura se resumen en la tabla 3 donde no se observa a penas diferencia entre las dos temperaturas elegidas.

Cuadro 3: Media de las métricas por temperatura

Temperatura	BLEU	ROUGE1	ROUGE2	ROUGEL	BERT_P	BERT_R	BERT_F1
0.2	0.082000	0.253000	0.082000	0.136000	0.700000	0.692000	0.695000
0.8	0.082000	0.263000	0.083000	0.137000	0.698000	0.691000	0.694000

En la Figura 2 se observan las métricas por modelo. Aunque en general obtienen resultados similares, el modelo que obtiene el mejor desempeño es **Phi-4**, seguido de **LLaMa3.2** y **Mistral**. Si bien actualmente no existe una gran cantidad de estudios sobre el uso de Phi-4 en el ámbito clínico, los resultados aquí obtenidos coinciden con estudios previos que reportan buen desempeño de Mistral (Jung et al., 2024) y LLaMA (Guo & Sarker, 2025) en tareas relacionadas con procesamiento de lenguaje biomédico.

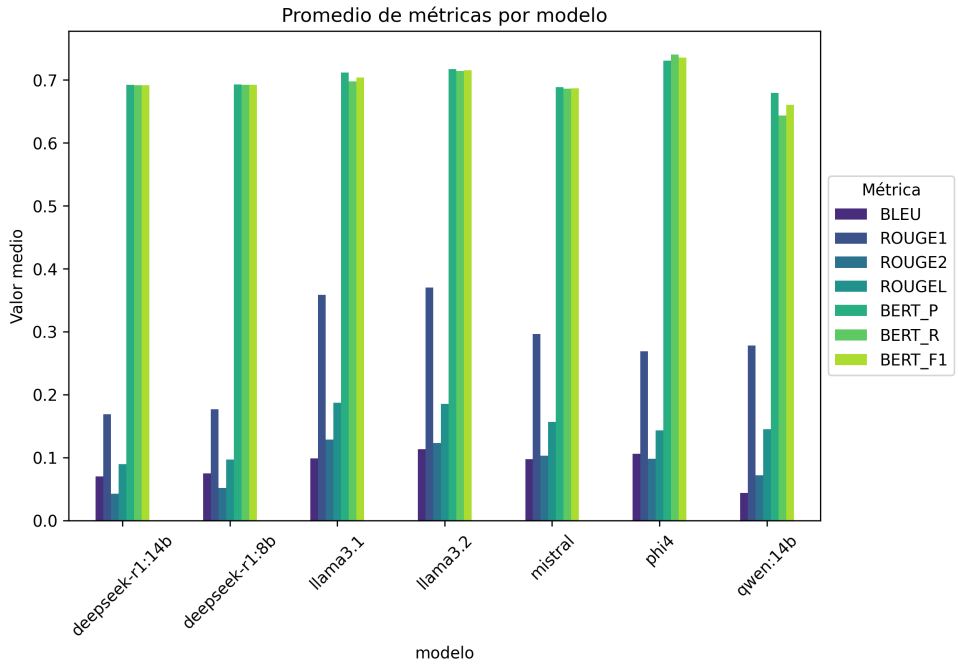

Figura 2: Métricas por modelo. Comparación de los modelos evaluados según las métricas BLEU, ROUGE y BERTScore.

En la figura 4 se observan las métricas por contexto, siendo bastante parecidas. La técnica de prompting de generación por partes sobresale con unos resultados ligeramente mejores en todas las métricas y modelos. Esto se debe a que al segmentar el historial de entrada, la ventana de contexto de los modelos no se ven desbordadas tan facilmente.

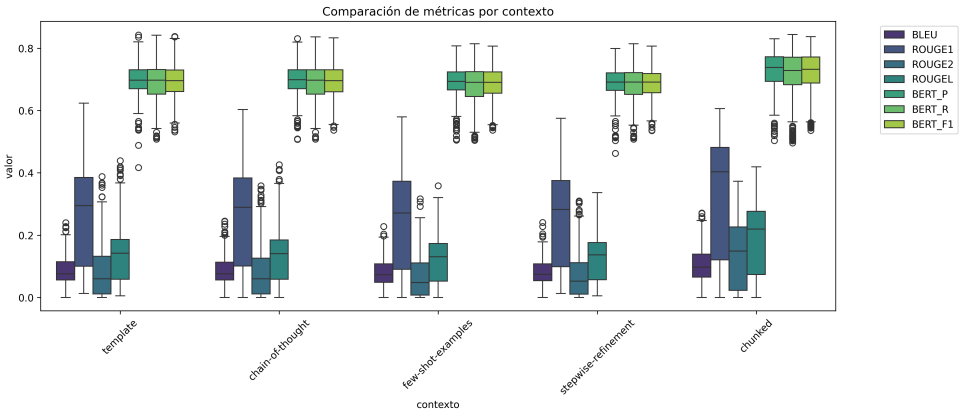

Figura 3: Métricas por contexto. Comparación de las métricas de similitud según el contexto utilizado.

En las figuras 4 y 5 se observan las métricas por modelo y contexto. Destaca el modelo phi4 y mistral, la estrategia de prompt por partes es la que mejor funciona con estos modelos. Parece que los modelos con ventanas de contexto más grandes como los destilados de deepseek no se benefician tanto de la segmentación del historial.

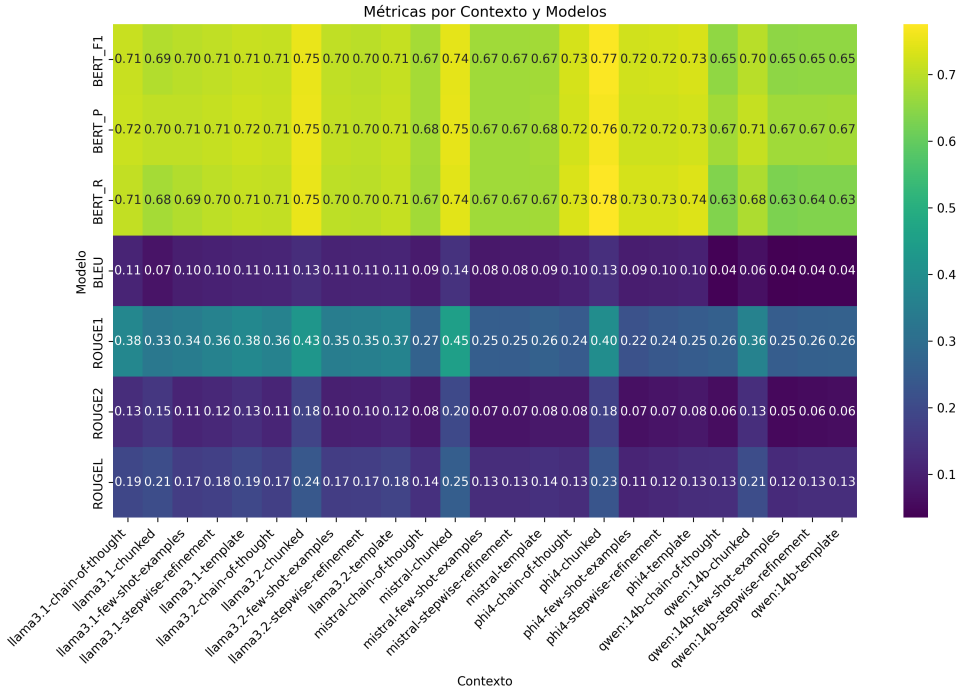

Figura 4: Métricas por contexto y modelos. Comparación de las métricas de similitud para los distintos modelos bajo los mismos contextos.

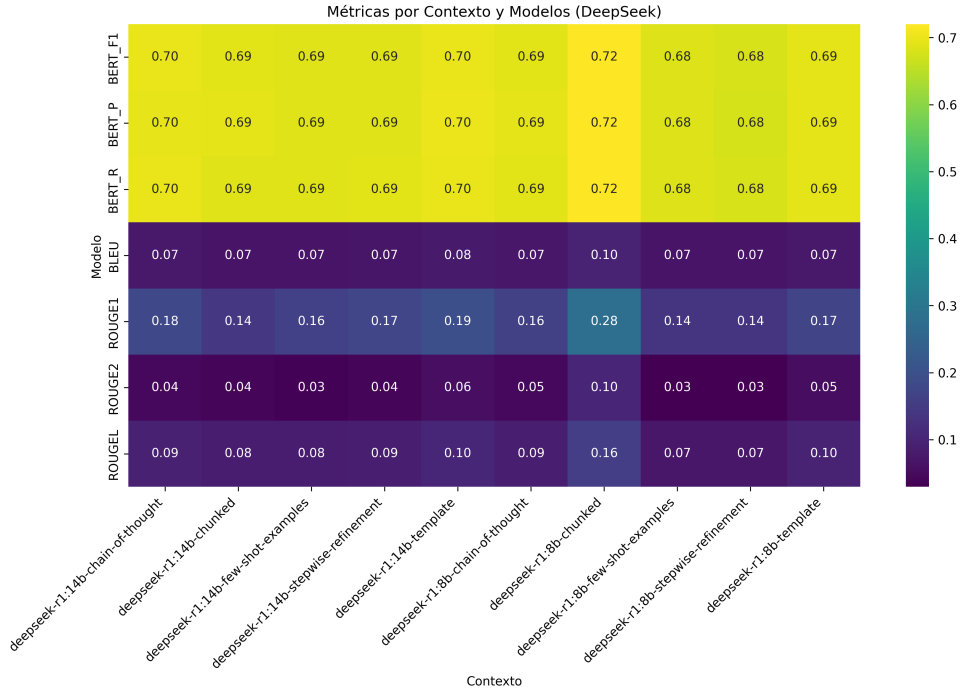

Figura 5: Métricas por contexto y Deepseek. Comparación de las métricas de similitud para el modelo Deepseek bajo diferentes contextos.

4.2.2. Tiempo de inferencia

En cuanto a la complejidad temporal, se midió el tiempo de inferencia de cada uno de los modelos evaluados para determinar su eficiencia en la ejecución.

En la tabla 4 se observa como hay una gran variación en el tiempo de inferencia, siendo los modelos destilados por deepseek varias veces mayores a los base. El mayor tiempo medio de inferencia es el destilado de qwen por deepseek siendo 386 segundos y el menor llama3.2 con 18 segundos. Teniendo en cuenta que al hacer una inferencia por partes el tiempo necesario es el tiple, se decidió que phi4 seguía siendo el candidato ideal, ya que se prioriza la exactitud del modelo que el tiempo de respuesta.

Cuadro 4: Tiempo de ejecución por modelo

Modelo	Media	Desviación
deepseek-r1:8b	157.26	85.74
deepseek-r1:14b	386.24	182.28
llama3.1	48.2	37.64
llama3.2	18.2	3.23
qwen:14b	42.61	2.65
mistral	60.98	60.25
phi4	172.79	47

4.3. Implementación de modelo final

4.3.1. RAG

En la Tabla 5 se presentan las métricas obtenidas por el modelo Phi-4 utilizando generación por partes junto con el sistema RAG multipregunta. El sistema alcanzó resultados sólidos en calidad de resumen, ligeramente mayores a los resultados sin RAG, con tiempos de ejecución de 523 segundos. Este valor incluye el tiempo requerido para realizar la búsqueda en la base de datos vectorial y ejecutar tres inferencias por segmento del historial clínico. Aunque el proceso implica un mayor coste computacional, la arquitectura permite que algunas fases, como la inferencia inicial de ideas clave, puedan ser ejecutadas en paralelo, lo cual abre la posibilidad de optimizar el rendimiento en futuras implementaciones.

Se ha confirmado que el sistema RAG es capaz de obtener información de contexto útil de la fuente de datos, no obstante, sería óptimo que en esta fuente de datos encontrásemos historias clínicas resumidas en lugar de ptreguntas y respuestas, de este modo el modelo podría encontrar contexto de cómo se suele resumir una parte del historial clínico con varios ejemplos.

Cuadro 5: Métricas obtenidas por RAG

	BLEU	**ROUGE1**	**ROUGE2**	**ROUGEL**	**BERT_P**	**BERT_R**	**BERT_F1**
Media	0.146019	0.475099	0.213789	0.259569	0.786528	0.792976	0.789614
Desv. típica	0.035366	0.088174	0.072935	0.063156	0.040259	0.028887	0.034523

4.3.2. Ajuste fino del modelo

Tras llevar a cabo el ajuste fino LoRA del modelo llama3.2:3b-Instruct utilizando el conjunto de datos adaptado de ruslanmv/ai-medical-chatbot (Vsevolodovna, 2024), se observaron cambios significativos en el estilo y contenido de las respuestas generadas. En particular, el modelo ajustado tiende a replicar el patrón de respuesta del corpus de entrenamiento, incluso cuando no se proporciona un contexto explícito. Un ejemplo de ello se muestra a continuación, donde el modelo responde con una estructura y tono casi idénticos al texto original con el que fue entrenado:

> *Hola, gracias por tu pregunta. Después de revisar la información proporcionada, puedo decir que la paciente tiene una buena tolerancia al tratamiento con abemaciclib. [...] En resumen, la paciente tiene una buena tolerancia al tratamiento con abemaciclib y no hay efectos adversos graves en la función hepática o renal.*

Este tipo de respuesta refleja directamente el estilo aprendido del dataset de entrenamiento, cuyo tono es conversacional y orientado a asesoramiento médico general:

> *Hi there Acne has multifactorial etiology. Only acne soap does not improve if ypu have grade 2 or more grade acne. You need to have oral and topical medications. This before writing medicines i need to confirm your grade of acne...*

Por contraste, antes del ajuste fino, el modelo era capaz de adaptarse mejor al formato deseado en el contexto, generando salidas más estructuradas y acordes al estilo clínico esperado. Por ejemplo:

> *Resumen del Registro Médico*
> *Historial Médico Personal y Familiar: El paciente es de 77 años de edad. Su padre fue diagnosticado con cáncer de cavidad oral a los 50 años...*

Estos resultados indican que, si bien el modelo efectivamente aprendió de los datos durante el ajuste fino, la influencia del estilo de respuesta original del conjunto de datos fue excesiva y perjudicial para el objetivo final del proyecto, que requería respuestas estructuradas y en un formato médico específico. En consecuencia, se concluye que el fine-tuning fue exitoso en términos de aprendizaje, pero inadecuado en cuanto a la alineación con el estilo y formato requeridos para el sistema final.

4.4. Desarrollo del prototipo de aplicación web

Como parte de la validación práctica del sistema, se comprobó el funcionamiento de la aplicación web desarrollada. Esta permite introducir historiales clínicos reales y obtener resúmenes generados automáticamente mediante el modelo seleccionado.

En la Figura 6 se muestra la interfaz de la aplicación, que facilita la interacción con el sistema de forma sencilla.

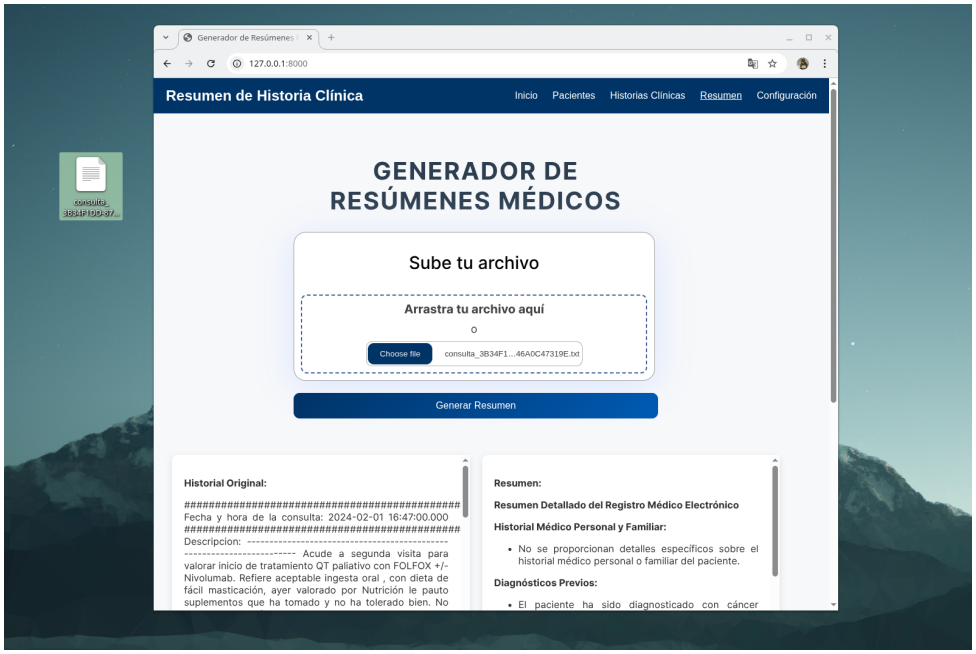

Figura 6: Interfaz gráfica de la aplicación demostrativa

5

Conclusiones y Líneas Futuras

5.1. Conclusiones y Líneas Futuras

5.1.1. Conclusiones

Este trabajo ha demostrado que los modelos de lenguaje pequeños (SLMs) pueden generar resúmenes clínicos de calidad aceptable mediante el uso de estrategias de *prompting* adecuadas, siendo especialmente efectiva la generación por partes. Aunque las métricas de similitud no igualan a modelos de mayor escala como GPT-4o-mini, se han obtenido resultados prometedores, destacando modelos como phi4 y llama3.2.

La implementación del enfoque de RAG con estrategia multipregunta ha permitido mejorar la calidad de los resúmenes generados, si bien a costa de un aumento en el tiempo de inferencia. Sin embargo, la arquitectura utilizada es fácilmente paralelizable, lo cual sugiere oportunidades de optimización para futuras implementaciones.

Respecto al ajuste fino del modelo llama3.2, los resultados muestran un aprendizaje eficaz, pero también un sobreajuste al estilo conversacional del conjunto de datos utilizado. Esto afectó negativamente la estructura clínica esperada en los resúmenes, lo que evidencia la necesidad de un corpus mejor alineado con nuestro problema.

En conjunto, se ha validado un flujo de trabajo reproducible para evaluar y optimizar modelos ligeros en tareas de síntesis clínica, combinando evaluación automática y validación por parte de expertos médicos.

Se ha desarrollado una aplicación web funcional que permite ingresar historiales clínicos y obtener resúmenes generados automáticamente. Esta herramienta facilita la validación práctica de los modelos y demuestra la aplicabilidad real del sistema en entornos clínicos.

5.1.2. Líneas Futuras

- **Reentrenamiento con corpus clínico estructurado:** Se propone realizar un nuevo ajuste fino utilizando conjuntos de datos más alineados con el lenguaje y formato de historia y resúmen clínico, evitando estilos conversacionales.

- **Optimización del sistema RAG:** Es recomendable paralelizar las etapas de recuperación y generación de contexto para reducir el tiempo de inferencia. También se podrían explorar técnicas de prefiltrado de recuperación para minimizar el volumen de consultas.

- **Evaluación cualitativa ampliada:** Se sugiere incluir a un mayor número de profesionales médicos en la validación de los resultados.

- **Integración en entornos hospitalarios:** Finalmente, se plantea la posibilidad de desplegar el sistema en entornos clínicos reales o simulados, mediante su conexión a sistemas de historia clínica electrónica (HCE), para evaluar su impacto práctico y su aceptación por parte de los usuarios finales.

- **Desarrollo de un asistente clínico inteligente:** Inspirado en herramientas como *Dragon Copilot* de Microsoft, que asiste a los médicos en la documentación clínica mediante dictado por voz, generación automática de notas y acceso a información médica relevante en tiempo real ("Conozcan Microsoft Dragon Copilot: su nuevo asistente de IA para el flujo de trabajo clínico", 2025), se plantea el desarrollo de un asistente similar adaptado al contexto local. Este asistente tendría como objetivo asistir a los profesionales de la salud en la redacción y revisión de historiales clínicos, proporcionando resúmenes automáticos, sugerencias de documentación y alertas sobre inconsistencias o información faltante, todo ello integrado en los sistemas de historia clínica electrónica (HCE).

Referencias

Alkhalaf, M., Yu, P., Yin, M., & Deng, C. (2024). Applying generative AI with retrieval augmented generation to summarize and extract key clinical information from electronic health records. *Journal of Biomedical Informatics*, *156*, 104662. https://doi.org/https://doi.org/10.1016/j.jbi.2024.104662

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

Chopra, A., Prashar, A., & Sain, C. (2013). Natural language processing. *International journal of technology enhancements and emerging engineering research*, *1*(4), 131-134.

Conozcan Microsoft Dragon Copilot: su nuevo asistente de IA para el flujo de trabajo clínico [Accedido el 11 de mayo de 2025]. (2025).

Cyganek, B., Graña, M., Krawczyk, B., Kasprzak, A., Porwik, P., Walkowiak, K., & Woźniak, M. (2016). A survey of big data issues in electronic health record analysis. *Applied Artificial Intelligence*, *30*(6), 497-520.

Davis, J., Sounack, T., Sciacca, K., Brain, J. M., Durieux, B. N., Agaronnik, N. D., & Lindvall, C. (2025). MedSlice: Fine-Tuned Large Language Models for Secure Clinical Note Sectioning. https://arxiv.org/abs/2501.14105

Face, H. (2023). Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA. https://huggingface.co/blog/4bit-transformers-bitsandbytes

Face, H. (2025). Templates - Hugging Face Transformers. https://huggingface.co/docs/transformers/main/en/chat_templating

Frey, L., Lenert, L., & Lopez-Campos, G. (2014). EHR big data deep phenotyping. *Yearbook of medical informatics*, *23*(01), 206-211.

Gillum, R. F. (2013). From papyrus to the electronic tablet: a brief history of the clinical medical record with lessons for the digital age. *The American journal of medicine*, *126*(10), 853-857.

Grail, Q., Perez, J., & Gaussier, E. (2021). Globalizing BERT-based transformer architectures for long document summarization. En P. Merlo, J. Tiedemann & R. Tsarfaty (Eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (pp. 1792-1810). Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.154

Guluzade, A., Heiba, N., Boukhers, Z., Hamiti, F., Polash, J. H., Mohamad, Y., & Velasco, C. A. (2025). ELMTEX: Fine-Tuning Large Language Models for Structured Clinical Information Extraction. A Case Study on Clinical Reports. https://arxiv.org/abs/2502.05638

Guo, Y., & Sarker, A. (2025). Benchmarking Open-Source Large Language Models on Healthcare Text Classification Tasks. https://arxiv.org/abs/2503.15169

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

Jung, H., Kim, Y., Choi, H., Seo, H., Kim, M., Han, J., Kee, G., Park, S., Ko, S., Kim, B., Kim, S., Jun, T. J., & Kim, Y.-H. (2024). Enhancing Clinical Efficiency through LLM: Discharge Note Generation for Cardiac Patients. https://arxiv.org/abs/2404.05144

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, *33*, 9459-9474.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. https://arxiv.org/abs/2005.11401

Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Text Summarization Branches Out*, 74-81. https://aclanthology.org/W04-1013/

Moreno-Barea, F. J., López-García, G., Mesa, H., Ribelles, N., Alba, E., Jerez, J. M., & Veredas, F. J. (2025). Named entity recognition for de-identifying Spanish electronic health records. *Computers in Biology and Medicine*, *185*, 109576. https://doi.org/https://doi.org/10.1016/j.compbiomed.2024.109576

OpenAI. (2024). Batch Processing - OpenAI API Documentation [Accedido el 17 de abril de 2025]. https://platform.openai.com/docs/guides/batch

Pal, A., Umapathi, L. K., & Sankarasubbu, M. (2022). MedMCQA: A Large-scale Multi-Subject Multi-Choice Dataset for Medical domain Question Answering. En G. Flores, G. H. Chen, T. Pollard, J. C. Ho & T. Naumann (Eds.), *Proceedings of the Conference on Health, Inference, and Learning* (pp. 248-260, Vol. 174). PMLR. https://proceedings.mlr.press/v174/pal22a.html

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. En P. Isabelle, E. Charniak & D. Lin (Eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311-318). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Parnami, A., & Lee, M. (2022). Learning from Few Examples: A Summary of Approaches to Few-Shot Learning. https://arxiv.org/abs/2203.04291

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., & Lewis, M. (2023). Measuring and Narrowing the Compositionality Gap in Language Models. https://arxiv.org/abs/2210.03350

Ren, Y., & Sutherland, D. J. (2024). Learning dynamics of llm finetuning. *arXiv preprint arXiv:2407.10490*.

Ryu, S., Do, H., Kim, Y., Lee, G. G., & Ok, J. (2024). Key-Element-Informed sLLM Tuning for Document Summarization. https://arxiv.org/abs/2406.04625

Saba, W., Wendelken, S., & Shanahan, J. (2024). Question-Answering Based Summarization of Electronic Health Records using Retrieval Augmented Generation. https://arxiv.org/abs/2401.01469

Searle, T., Ibrahim, Z., Teo, J., & Dobson, R. (2021). Estimating redundancy in clinical text. *Journal of Biomedical Informatics*, *124*, 103938. https://doi.org/10.1016/j.jbi.2021.103938

Sreenivas, S. T., Muralidharan, S., Joshi, R. B., Chochowski, M., Patwary, M., Molchanov, P., Shoeybi, M., Kautz, J., Mahabaleshwarkar, A. S., Shen, G., et al. (2025). LLM Pruning and Distillation in Practice. *ICLR*.

Sun, S., Yuan, R., Cao, Z., Li, W., & Liu, P. (2024). Prompt Chaining or Stepwise Prompt? Refinement in Text Summarization. https://arxiv.org/abs/2406.00507

Team, U. (2025). Fine-tuning Guide - Unsloth Documentation. https://docs.unsloth.ai/get-started/fine-tuning-guide

Tinn, R., Cheng, H., Gu, Y., Usuyama, N., Liu, X., Naumann, T., Gao, J., & Poon, H. (2023). Fine-tuning large neural language models for biomedical natural language processing. *Patterns*, *4*(4), 100729. https://doi.org/https://doi.org/10.1016/j.patter.2023.100729

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vsevolodovna, R. M. (2024). AI Medical Chatbot [Repositorio de código abierto para un chatbot médico basado en inteligencia artificial].

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. https://arxiv.org/abs/2201.11903

Zhang, G., Fukuyama, K., Kishimoto, K., & Kuroda, T. (2024). Optimizing Automatic Summarization of Long Clinical Records Using Dynamic Context Extension:Testing and Evaluation of the NBCE Method. https://arxiv.org/abs/2411.08586

Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., & Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT. https://arxiv.org/abs/1904.09675

Zhang, Y., Jin, H., Meng, D., Wang, J., & Tan, J. (2025). A Comprehensive Survey on Process-Oriented Automatic Text Summarization with Exploration of LLM-Based Methods. https://arxiv.org/abs/2403.02901

Zhao, B., Zan, H., Niu, C., Chang, H., & Zhang, K. (2024). Automatic Generation of Discharge Summary of EMRs Based on Multi-granularity Information Fusion. En H. Xu, Q. Chen, H. Lin, F. Wu, L. Liu, B. Tang, T. Hao & Z. Huang (Eds.), *Health Information Processing* (pp. 254-269). Springer Nature Singapore.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*, *1*(2).

Apéndice A
Manual de Instalación

Apéndice B
Código

Recomendacion del tutor de explicar un poco todo el codigo aqui

B.1. Generacion de Resúmenes Modelo

```
import os
import json

def generar_jsonl(contextos_dir, entradas_dir, jsonl_file):
    with open(jsonl_file, 'w', encoding='utf-8') as jsonl:
        for contexto_nombre in os.listdir(contextos_dir):
            if contexto_nombre.endswith('.txt'):
                with open(os.path.join(contextos_dir,
  contexto_nombre), 'r', encoding='utf-8') as ctx_file:
                    contexto = ctx_file.read()

                for archivo_nombre in os.listdir(entradas_dir):
                    if archivo_nombre.endswith('.txt'):
                        with open(os.path.join(entradas_dir,
  archivo_nombre), 'r', encoding='utf-8') as entrada_file:
                            contenido = entrada_file.read()

                        # Crear una solicitud para cada
  combinación de contexto y archivo
                        solicitud = {
                            "custom_id": f"{archivo_nombre}
  _with_{contexto_nombre}",
                            "method": "POST",
```

```
                            "url": "/v1/chat/completions",
                            "body": {
                                "model": "gpt-4o-mini",
                                "messages": [
                                    {"role": "system", "content
    ": contexto},
                                    {"role": "user", "content":
     f"<record>\n{contenido}"}
                                ],
                                "max_tokens": 500
                            }
                        }
                        jsonl.write(json.dumps(solicitud) + '\n
    ')

    print(f"Archivo {jsonl_file} generado para Batch API.")

if __name__ == "__main__":
    common_data_dir = '../../common_data'
    contextos_dir = os.path.join(common_data_dir, 'contextos')
    entradas_dir = os.path.join(common_data_dir, 'entradas')
    jsonl_file = 'batch_input.jsonl'
    generar_jsonl(contextos_dir, entradas_dir, jsonl_file)

    print("JSONL generado con éxito.")
```

```
import os
from openai import OpenAI

def obtener_api():
    openai = OpenAI()
```

```
    api_key = os.getenv('OPENAI_API_KEY')
    if not api_key:
        raise ValueError("Error: No se encontró la clave de la
   API. Configura la variable de entorno 'OPENAI_API_KEY'.")
    return openai

def subir_batch(jsonl_file):
    openai = obtener_api()

    # Subir archivo a Batch API
    batch_input_file = openai.files.create(
        file=open(jsonl_file, 'rb'),
        purpose='batch'
    )

    print(f"Archivo de entrada subido con ID: {batch_input_file
   .id}")

    # Crear un trabajo batch
    batch_job = openai.batches.create(
        input_file_id=batch_input_file.id,
        endpoint="/v1/chat/completions",
        completion_window="24h",
        metadata={"description": "Resumen de historiales
   clínicos prompt templates"}
    )

    print(f"Trabajo batch creado con ID: {batch_job.id}")
    return str(batch_job)
# Subir batch y guardar respuesta
if __name__ == "__main__":
```

```
    jsonl_file = 'batch_input.jsonl'
    batch_job = subir_batch(jsonl_file)

    with open('batch_job_id.txt', 'w') as batch_job_file:
        batch_job_file.write(batch_job)

    #  File "/home/alex/Documents/UNI/Cuarto/segundo_cuatri/tfg
   /code/gpt_api_EHR_summarization/subir_batch.py", line 38, in
    <module>
    #batch_job_file.write(batch_job)
    #TypeError: write() argument must be str, not Batch
    print(f"Trabajo batch subido con éxito. ID: {batch_job.id}"
   )
```

```
import os
from openai import OpenAI

def obtener_api():
    openai = OpenAI()
    api_key = os.getenv('OPENAI_API_KEY')
    if not api_key:
        raise ValueError("Error: No se encontró la clave de la
   API. Configura la variable de entorno 'OPENAI_API_KEY'.")
    return openai

def estado_batch(batch_job_id):
    openai = obtener_api()

    # Recuperar el estado del trabajo batch
    batch_job = openai.batches.retrieve(batch_job_id)
    #guardamos el estado del batch en un archivo
```

```
    return batch_job

batch_job_id = 'batch_67d87bc4e99c8190b1b9dc00cf6e7c9b'
with open('batch_job_status.txt', 'w') as batch_job_status_file
   :
        batch_job_status_file.write(str(estado_batch(
   batch_job_id)))

print(f"Estado del trabajo batch guardado con éxito en
   batch_job_status.txt.")
```

```
import os
from openai import OpenAI

def obtener_api():
    openai = OpenAI()
    api_key = os.getenv('OPENAI_API_KEY')
    if not api_key:
        raise ValueError("Error: No se encontró la clave de la
   API. Configura la variable de entorno 'OPENAI_API_KEY'.")
    return openai

def recuperar_batch(file_name):
    openai = obtener_api()
    file_response = openai.files.content(file_name)
    return file_response.text

if __name__ == "__main__":
    file_name = 'file-PWeoYjCEL9az9Q3djxosY7'
    with open('respuesta_batch.json', 'w') as
```

```
    batch_response_file:
        batch_response_file.write(recuperar_batch(file_name))
```

```
import os
import json

json_entrada = 'respuesta_batch.json'
carpeta_salida = 'archivos'
with open(json_entrada, 'r') as file:
    while linea:=file.readline():
        data = json.loads(linea)
        with open(f'{carpeta_salida}/respuesta_{data["custom_id
    "]}', 'w') as f:
            f.write(str(data["response"]["body"]["choices"][0][
    "message"]["content"]))

print('Proceso terminado')
```

Apéndice C
Figuras

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA