

Universidad Internacional de La Rioja

Facultad de Ciencias de la Salud

Máster Universitario en Bioinformática

Desarrollo de un algoritmo predictivo para la personalización de estilos de vida basado en genómica y biométrica

|  |  |
| --- | --- |
| Trabajo fin de Estudio presentado por: | Maximiliano Hernández Sahuquillo y David Fernández Martín |
| Tipo de trabajo: | 2 |
| Director/a: | Dr. José Arturo Mora Soto |
| Fecha: | 14 de marzo de 2024 |

Resumen

El presente TFM tiene como objetivo desarrollar un algoritmo predictivo para la personalización de estilos de vida basado en datos genómicos y biométricos. El algoritmo estará diseñado para generar recomendaciones personalizadas de estilo de vida para cada individuo, teniendo en cuenta su predisposición genética a ciertas enfermedades, su respuesta al ejercicio y su respuesta a diferentes tipos de dieta.

**Palabras clave:** estilo de vida, genómica, biométrica

Abstract

This TFM aims to develop a predictive algorithm for lifestyle personalisation based on genomic and biometric data. The algorithm will be designed to generate personalised lifestyle recommendations for each individual, taking into account their genetic predisposition to certain diseases, their response to exercise and their response to different types of diet.

**Keywords**: lifestyle, genomics, biometrics

Índice de contenidos

[1. Introducción 9](#_Toc171027604)

[1.1. Justificación 9](#_Toc171027605)

[1.2. Planteamiento del problema 12](#_Toc171027606)

[1.3. Objetivos 13](#_Toc171027607)

[2. Marco teórico 14](#_Toc171027608)

[2.1. Large Language Models (LLMs) en el ámbito médico 14](#_Toc171027609)

[2.1.1. Introducción a los LLMs en el ámbito médico 14](#_Toc171027610)

[2.1.2. La tabla de clasificación Open Medical-LLM 15](#_Toc171027611)

[2.1.3. Med-PALM 2: Un modelo de lenguaje large de última generación para el ámbito médico 15](#_Toc171027612)

[2.2. Chatbots especializados en medicina 16](#_Toc171027613)

[2.3. Manejo de hallazgos secundarios en estudios genómicos 17](#_Toc171027614)

[3. Metodología 18](#_Toc171027615)

[3.1. Desarrollo del Chatbot basado en RAG 18](#_Toc171027616)

[3.2. Arquitectura y funcionamiento de la API 19](#_Toc171027617)

[3.3. Proceso de búsqueda de información 24](#_Toc171027618)

[3.3.1. Selección y procesamiento de documentos 24](#_Toc171027619)

[3.3.2. Integración en la base de conocimientos 25](#_Toc171027620)

[3.3.3. Búsqueda semántica y enlazado de prompt 27](#_Toc171027621)

[3.3.4. Integración de fuentes de información adicionales 28](#_Toc171027622)

[3.4. Evaluación y refinamiento del modelo 29](#_Toc171027623)

[3.4.1. Conjunto de datos de prueba 29](#_Toc171027624)

[3.4.2. Métricas de evaluación 30](#_Toc171027625)

[3.4.3. Análisis de rendimiento 31](#_Toc171027626)

[3.5. Desarrollo del prototipo de la interfaz 32](#_Toc171027627)

[3.5.1. Consideraciones para el desarrollo de la web 32](#_Toc171027628)

[3.5.2. Pasos para el desarrollo la API 33](#_Toc171027629)

[3.5.3. Pruebas de validación 38](#_Toc171027630)

[3.6. Esquema y funcionamiento formal del sistema 40](#_Toc171027631)

[3.6.1. Funcionamiento de main.py 41](#_Toc171027632)

[3.6.2. Funcionamiento del custom\_agent.py 43](#_Toc171027633)

[3.6.3. Funcionamiento del streamlit\_app.py 45](#_Toc171027634)

[3.6.4. Flujo de la aplicación 46](#_Toc171027635)

[4. Resultados y discusión 50](#_Toc171027636)

[4.1. Resultados 50](#_Toc171027637)

[4.1.1. Pruebas previas mediante Jupyter notebooks 51](#_Toc171027638)

[4.1.2. Proceso de webscrapping mediante Beautifulsoup 53](#_Toc171027639)

[4.1.3. Implementación de un sistema de búsqueda genética basado en Langchain y Elasticsearch 56](#_Toc171027640)

[4.1.4. Desempeño final del chatbot 57](#_Toc171027641)

[4.1.5. Ejemplos de interacciones 60](#_Toc171027642)

[4.1.6. Estadísticas de uso 64](#_Toc171027643)

[4.1.7. Evaluación de la interfaz de usuario 65](#_Toc171027644)

[4.2. Discusión 66](#_Toc171027645)

[4.2.1. Interpretación de los resultados 66](#_Toc171027646)

[4.2.2. Comparación con otros chatbots médicos 67](#_Toc171027647)

[4.2.3. Limitaciones y desafíos 69](#_Toc171027648)

[4.2.4. Repercusiones prácticas de chatbots especializados 71](#_Toc171027649)

[4.2.5. Puntos fuertes y ventajas 72](#_Toc171027650)

[4.2.6. Implicaciones en la seguridad 73](#_Toc171027651)

[4.2.7. Implicaciones para la práctica clínica 75](#_Toc171027652)

[4.2.8. Direcciones futuras 76](#_Toc171027653)

[5. Conclusiones 78](#_Toc171027654)

[5.1.1. Contribuciones principales 79](#_Toc171027655)

[5.1.2. Perspectivas de desarrollo 79](#_Toc171027656)

[Referencias bibliográficas 81](#_Toc171027657)

Figura 1*. “Med-PaLM 2 performance on MultiMedQA Left: Med-PaLM 2 achieved an accuracy of 86.5% on USMLE-style questions in the MedQA dataset. Right: In a pairwise ranking study on 1066 consumer medical questions, Med-PaLM 2 answers were preferred over physician answers by a panel of physicians across eight of nine axes in our evaluation framework”. De Singhal K, Tu T, Gottweis J, Sayres R, Wulczyn E, Hou L, et al. Towards Expert-Level Medical Question Answering with Large Language Models [Internet]. arXiv; 2023 [citado 1 de junio de 2024]. Disponible en: http://arxiv.org/abs/2305.09617. (Licensed under CC BY 4.0) https://creativecommons.org/licenses/by/4.0/* 10

Figura 2. *Comparación del rendimiento de diferentes modelos de lenguaje en término de parámetros. Adaptado de (7)* 16

Figura 3*. Estructura del repositorio* 18

Figura 4. *Framework del RAG* 20

Figura 5. *Vectorización con OLLAMA y almacenamiento en FAISS* 26

Figura 6. *Procedencia de la información de DisGeNET* 36

Figura 7. *Clasificación de las preguntas* 44

Figura 8. *Código presente en custom\_agent.py* 48

Figura 9. *Código presene en streamlit.app* 49

Figura 10. *Código de ejemplo sobre el uso de webscrapping* 56

Figura 11. *Código de ejemplo sobre el uso de webscrapping* 57

Figura 12. *Carga de claves y establecimiento de Elasticsearch* 59

Figura 13. *Comparación del rendimiento de diferentes modelos de lenguaje en términos de precisión en diversos conjuntos de (7)* 60

Figura 14. *Ejemplo de una pregunta compleja sobre DBE en un entorno comunitario, junto con la respuesta generada por el chatbot. El contexto proporcionado proviene de un estudio publicado en Arxiv. (8).* 62

Figura 15. *Comparación del rendimiento de Med-PaLM 2 con otros modelos de lenguaje en diferentes conjuntos de datos. médicos. Adaptado de (8).* 70

Índice de tablas

Tabla 1. *Comparación del modelo desarrollado con respectos a otros existentes* 66

# **Introducción**

Para este proyecto de TFM, se ha desarrollado una API mediante el uso del conjunto de herramientas proporcionadas por FastAPI, que nos permite alcanzar nuestro objetivo principal: crear un sistema de personalización de estilos de vida basado en genómica y biometría. Esta API utiliza diversas bibliotecas y herramientas de procesamiento de lenguaje natural para generar recomendaciones personalizadas a partir de los datos proporcionados por los usuarios.

En las próximas páginas se introducirá este trabajo, primero se abordarán los motivos que llevaron a escoger esta temática, seguido de la introducción del problema a resolver y, por último, se el planteamiento del objetivo principal y objetivos secundarios a alcanzar.

## Justificación

En los últimos años, el campo de la inteligencia artificial ha experimentado un avance vertiginoso, especialmente en el procesamiento del lenguaje natural (NLP) y el desarrollo de modelos de lenguaje de gran tamaño (LLM) (1). Modelos como GPT-3, GPT-4 y Med-PaLM2 (2), están transformando la atención médica al demostrar habilidades excepcionales para entender y generar texto similar al humano, con creces el último nombrado, superando a todos los modelos con los que se realizaron las pruebas (Fig. 1). Estas capacidades los convierten en herramientas valiosas para abordar tareas médicas complejas, mejorando así la atención al paciente. Su potencial se ha destacado en aplicaciones como responder preguntas médicas, sistemas de diálogo y generación de texto. Con el aumento de registros médicos electrónicos y datos generados por pacientes, los LLM pueden ayudar a los profesionales de la salud a obtener información crucial y tomar decisiones informadas.

Gráfico

Descripción generada automáticamente

Figura 1*. “Med-PaLM 2 performance on MultiMedQA Left: Med-PaLM 2 achieved an accuracy of 86.5% on USMLE-style questions in the MedQA dataset. Right: In a pairwise ranking study on 1066 consumer medical questions, Med-PaLM 2 answers were preferred over physician answers by a panel of physicians across eight of nine axes in our evaluation framework”. De Singhal K, Tu T, Gottweis J, Sayres R, Wulczyn E, Hou L, et al. Towards Expert-Level Medical Question Answering with Large Language Models [Internet]. arXiv; 2023 [citado 1 de junio de 2024]. Disponible en: http://arxiv.org/abs/2305.09617. (Licensed under CC BY 4.0) https://creativecommons.org/licenses/by/4.0/*

Estos modelos han ganado notoriedad y presencia en el sector médico gracias a sus múltiples aplicaciones, desde el procesamiento de grandes volúmenes de datos hasta la interpretación de anotaciones y datos clínicos para generar informes adecuados.

En estudios recientes acerca del rendimiento de bots de chat basados en LLMs al responder preguntas médicas, han demostrado una impresionante cantidad de aciertos con respecto al conocimiento sobre bases de datos médicas. Esto es debido ya que en primer lugar, poseen suficientes conocimientos médicos para aplicaciones en medicina, lo que podría provocar un cambio de paradigma en la prestación de asistencia sanitario (1). Pero persiste una considerable inquietud sobre la preocupación de que los LLM puedan ser manipulados maliciosamente para generar contenidos dañinos o engañosos (3).

Ya que, a pesar de su promesa, los LLM enfrentan desafíos importantes en el ámbito médico. Mientras que los errores en usos recreativos tienen poco impacto, en el ámbito médico pueden tener consecuencias graves, incluso poniendo en peligro la vida de los pacientes. La precisión y confiabilidad de la información que proporcionan es crucial, ya que puede influir en decisiones médicas importantes, como diagnósticos y planes de tratamiento.

Otra facilidad aportada por estos modelos es que el proceso de cribado de la literatura (que se vuelve excesivamente laboriosa debido a que pueden existir miles de artículos con respecto a una patología o un gen) se vuelva más rápido y eficaz, permitiendo resumir la literatura científica y dejando a los investigadores y médicos la información de manera accesible con respecto al esfuerzo realizado (4).

A su vez en otros campos relacionados con la salud como la nutrición, numerosos estudios han contribuido al desarrollo de diversos métodos de recomendación de alimentos (5,6). Estos enfoques mediante el uso de LLM no sólo simplifican el proceso de elección de alimentos, sino que también promueven la adopción de hábitos sostenibles y saludables entre los usuarios. El poder de estas recomendaciones reside en su capacidad para integrar la ciencia de la nutrición en las necesidades de la población o de subpoblaciones, ofreciendo una guía intuitiva y práctica para las personas que desean tomar decisiones dietéticas informadas.

No obstante, los servicios convencionales de recomendación de alimentos orientados a la nutrición han encontrado limitaciones a la hora de comprender exhaustivamente la intrincada interacción entre la salud y el bienestar de las personas, que abarca parámetros fisiológicos, actividad física y calidad del sueño, y de alinearlos con necesidades nutricionales personalizadas. Concretamente, estos sistemas suelen tener dificultades para adaptar sus sugerencias alimentarias a las necesidades nutricionales específicas de cada persona. La variabilidad inherente a la forma en que la nutrición afecta a los individuos suscita preocupación por la ausencia de una auténtica personalización, además otra carencia importante es la falta de interactividad.

En la era de la medicina personalizada y la genómica, el manejo de hallazgos secundarios en estudios genómicos se ha convertido en un desafío significativo. La interpretación manual de estos hallazgos es un proceso laborioso y propenso a errores que requiere experiencia en genética médica y acceso a información actualizada. En este contexto, el desarrollo de herramientas bioinformáticas automatizadas basadas en LLM y técnicas de aprendizaje profundo emerge como una solución prometedora para agilizar y mejorar este proceso.

Este trabajo se enfoca en el desarrollo de un chatbot médico especializado que aprovecha los avances en procesamiento del lenguaje natural y aprendizaje automático para brindar respuestas precisas y confiables a preguntas médicas, con un énfasis particular en el manejo de hallazgos secundarios en estudios genómicos. A diferencia de simplemente utilizar un chatbot médico existente, este proyecto implica la creación de un sistema personalizado que integra múltiples fuentes de información médica y aplica técnicas avanzadas para generar respuestas adaptadas al contexto específico de la genómica y la medicina personalizada.

Finalmente, este Trabajo Fin de Máster tiene como objeto de estudio la implementación de un servicio sencillo e intuitivo, pero a la vez complejo, que permite a los usuarios acceder a una web para obtener recomendaciones personalizadas. Estas recomendaciones abarcan desde dudas sobre ciertas patologías hasta sugerencias de alimentos o cambios en función de distintos objetivos. Esto se realiza sin olvidar que la asistencia a un profesional de la salud será siempre necesaria.

## Planteamiento del problema

El objeto de estudio de este TFM, como se ha presentado en el apartado anterior es desarrollar un sistema que personalice los estilos de vida en función de la genética y la biometría, ya que recomendaciones generales sobre dieta y ejercicio son útiles, pero no siempre son efectivas para todas las personas.

## Objetivos

Como ya se ha adelantado, la propuesta consiste en desarrollar chatbot médico especializado basado en LLMs y técnicas de aprendizaje profundo para brindar respuestas precisas y confiables a preguntas médicas, con un enfoque particular en el manejo automático de hallazgos secundarios en estudios genómicos.

Los objetivos específicos incluyen:

1. Diseñar e implementar una arquitectura de sistema que integre LLMs, técnicas de procesamiento del lenguaje natural y aprendizaje profundo para el desarrollo del chatbot médico.
2. Desarrollar un proceso eficiente de búsqueda de información que combine la carga y vectorización de documentos médicos, búsqueda semántica y enlazado de prompts para generar respuestas precisas y contextualizadas.
3. Aplicar los algoritmos desarrollados para el chatbot al manejo automático de hallazgos secundarios en estudios genómicos, con el objetivo de agilizar su interpretación y facilitar la toma de decisiones clínicas informadas.
4. Evaluar exhaustivamente el desempeño del chatbot utilizando un conjunto de datos de prueba representativo, métricas de evaluación apropiadas y un análisis detallado del rendimiento.
5. Analizar los resultados obtenidos, comparar el desempeño del chatbot con otros sistemas similares, identificar limitaciones y desafíos, y discutir las implicaciones para la práctica clínica y las direcciones futuras de investigación.

# **Marco teórico**

No habría una fuente bibliográfica la cual haya significado una mayor influencia sobre cómo estructurar el entorno de trabajo o el proceso de realizar la web, sino que se ha ido obteniendo la información de distintas publicaciones u otras referencias.

## **Large Language Models (LLMs) en el ámbito médico**

### Introducción a los LLMs en el ámbito médico

Los modelos de lenguaje large (LLM, por sus siglas en inglés) han demostrado un inmenso potencial para revolucionar diversos aspectos de la atención sanitaria en los últimos años (7). Modelos como GPT-3, GPT-4 y Med-PaLM 2 han exhibido capacidades notables para comprender y generar texto similar al humano, convirtiéndolos en herramientas valiosas para abordar tareas médicas complejas y mejorar la atención al paciente (7). Estos modelos resultan especialmente prometedores en aplicaciones como la respuesta a preguntas médicas, los sistemas de diálogo y la generación de textos (7). Además, dado el crecimiento exponencial de los registros médicos electrónicos, la literatura médica y los datos generados por los pacientes, los LLM podrían ayudar a los profesionales sanitarios a extraer información valiosa y tomar decisiones informadas (7).

### La tabla de clasificación Open Medical-LLM

La tabla de clasificación Open Medical-LLM tiene como objetivo rastrear, clasificar y evaluar el rendimiento de los modelos de lenguaje grande (LLM) en tareas de respuesta a preguntas médicas (7). Evalúa los LLM en una amplia gama de conjuntos de datos médicos, incluyendo MedQA (USMLE), PubMedQA, MedMCQA y subconjuntos de MMLU relacionados con medicina y biología (7). La tabla ofrece una evaluación integral de los conocimientos médicos y las capacidades de respuesta a preguntas de cada modelo (7).

Los conjuntos de datos cubren diversos aspectos de la medicina, como conocimientos médicos generales, conocimientos clínicos, anatomía, genética y más (7). Contienen preguntas abiertas y de opción múltiple que requieren razonamiento y comprensión médica (7). La principal métrica de evaluación utilizada es la precisión (ACC) (7).

La plataforma va a permitir a los distintos profesionales e investigadores la evaluación de las ventajas y desventajas de los distintos LLMs, así escogiendo y ajustándolo a las necesidades específicas.

Tanto los modelos comerciales como los de código abierto obtienen un buen rendimiento en tareas como comprensión y razonamiento acerca de la literatura biomédica y con respecto a la toma de decisiones en relación con la aplicación de conocimientos clínicos. Los modelos comerciales como GPT-4 o MedPaLM-2 son capaces de alcanzar altas puntuaciones con relación a la precisión en conjuntos de datos médicos, pero los de código abierto como gemma-7b o Mistral-7B-v0.1 entre otros son capaces de mostrar un rendimiento competitivo a pesar de tener menor tamaño.

### Med-PALM 2: Un modelo de lenguaje large de última generación para el ámbito médico

Med-PaLM 2 es un modelo de lenguaje grande desarrollado específicamente para el dominio médico, que aprovecha una combinación de mejoras en el LLM base (PaLM 2), ajuste fino en el dominio médico y estrategias de prompting, incluyendo un novedoso enfoque de refinamiento por ensamblado (2).

Med-PaLM 2 obtuvo hasta un 86,5% en el conjunto de datos MedQA, mejorando Med-PaLM en más de un 19% y estableciendo un nuevo estado del arte (2). También se observó un rendimiento cercano o superior al estado del arte en los conjuntos de datos MedMCQA, PubMedQA y MMLU de temas clínicos (2).

Gráfico, Gráfico de burbujas

Descripción generada automáticamente

Figura 2. *Comparación del rendimiento de diferentes modelos de lenguaje en término de parámetros. Adaptado de (7)*

En detalladas evaluaciones humanas sobre preguntas de formato largo en múltiples ejes relevantes para aplicaciones clínicas, los médicos prefirieron las respuestas de Med-PaLM 2 a las producidas por médicos en ocho de cada nueve ejes relativos a la utilidad clínica (p < 0,001) (2). También se observaron mejoras significativas en comparación con Med-PaLM en cada eje de evaluación (p < 0,001) en conjuntos de datos de preguntas "adversas" de formato largo recientemente introducidos para sondear las limitaciones de los LLM (2).

Si bien se necesitan más estudios para validar la eficacia de estos modelos en entornos del mundo real, estos resultados destacan el rápido progreso hacia un rendimiento a nivel de médico en la respuesta a preguntas médicas (2).

## **Chatbots especializados en medicina**

Los chatbots médicos han surgido como una aplicación prometedora de los LLM en el ámbito de la salud. Estos asistentes virtuales están diseñados específicamente para interactuar con usuarios en un contexto médico, brindando información, orientación y apoyo personalizado (8). A diferencia de los chatbots genéricos, los chatbots médicos se basan en conocimientos especializados y están adaptados para manejar preguntas y situaciones relacionadas con la salud.

Sin embargo, el desarrollo de chatbots médicos efectivos y confiables presenta varios desafíos. Garantizar la precisión y la calidad de la información proporcionada es esencial, ya que los errores o la información engañosa pueden tener consecuencias graves para la salud de los pacientes (9). Además, los chatbots deben ser capaces de manejar la complejidad y la variabilidad del lenguaje médico, así como comprender el contexto y las sutilezas de las preguntas de los usuarios.

La implementación de inteligencia artificial en chatbots médicos también requiere una supervisión continua para asegurar su actualización y alineación con las últimas investigaciones y directrices médicas. Esto implica la colaboración estrecha con profesionales de la salud para validar y revisar periódicamente los algoritmos y bases de datos utilizados por los chatbots. La integración de aprendizaje automático permite que estos sistemas mejoren con el tiempo a través de la retroalimentación y los datos de interacción de los usuarios, pero esto también plantea la necesidad de manejar adecuadamente cuestiones de privacidad y seguridad de los datos.

Además, la aceptación y confianza del usuario en estos sistemas es crucial para su efectividad. Los chatbots deben ser diseñados para ofrecer una experiencia de usuario amigable, empática y comprensiva, lo que puede ayudar a reducir la ansiedad de los pacientes y mejorar la adherencia a los tratamientos. La personalización, basada en el historial médico y las necesidades individuales de los pacientes, también puede aumentar la eficacia de estos asistentes virtuales, proporcionando recomendaciones más precisas y relevantes.

## **Manejo de hallazgos secundarios en estudios genómicos**

Los avances en las tecnologías de secuenciación de nueva generación (NGS) han permitido la realización de estudios genómicos a gran escala, lo que ha llevado al descubrimiento de una gran cantidad de información genética relevante para la salud de los individuos (10). Además de los hallazgos primarios relacionados con la condición o enfermedad de interés, estos estudios a menudo revelan hallazgos secundarios, que son variantes genéticas asociadas con riesgos para la salud, implicaciones reproductivas o respuestas a medicamentos (11).

El manejo de hallazgos secundarios en estudios genómicos plantea desafíos significativos. La interpretación manual de estas variantes es un proceso complejo y laborioso que requiere experiencia en genética médica y acceso a información actualizada sobre la patogenicidad de las variantes (12). Además, la cantidad de datos generados por los estudios genómicos puede ser abrumadora, lo que dificulta la identificación y priorización de las variantes clínicamente relevantes.

Para abordar estos desafíos, se han desarrollado diversas herramientas bioinformáticas que buscan automatizar y optimizar el proceso de interpretación de hallazgos secundarios. Estas herramientas suelen basarse en algoritmos de aprendizaje automático y técnicas de procesamiento del lenguaje natural para analizar grandes volúmenes de datos genómicos y bibliográficos (13). Su objetivo es identificar variantes genéticas potencialmente patogénicas y proporcionar información sobre su relevancia clínica, lo que facilita la toma de decisiones informadas por parte de los profesionales de la salud.

El uso de herramientas bioinformáticas también ha permitido el avance de la medicina personalizada. Al integrar datos genómicos con información clínica y de estilo de vida, estas herramientas pueden predecir con mayor precisión cómo un paciente responderá a ciertos tratamientos. Esto es especialmente útil en el campo de la oncología, donde la identificación de mutaciones específicas en tumores puede guiar la selección de terapias dirigidas, mejorando significativamente los resultados del tratamiento (11).

Además, la implementación de bases de datos genéticas globales y consorcios de investigación ha facilitado la colaboración internacional, permitiendo una mejor comprensión de la variabilidad genética en diferentes poblaciones. Esto no solo ayuda a identificar variantes genéticas comunes y raras asociadas con enfermedades, sino que también promueve el desarrollo de guías clínicas basadas en evidencias sólidas y diversidad genética. La combinación de NGS, herramientas bioinformáticas avanzadas y esfuerzos colaborativos internacionales está revolucionando la genética médica y promete mejorar considerablemente la precisión y eficacia de los diagnósticos y tratamientos.

# **Metodología**

Antes de proceder con las diferentes secciones de esta parte, es importante destacar que, para una representación óptima y un seguimiento adecuado de su funcionalidad, todos los archivos, documentos y materiales complementarios mencionados estarán disponibles en <https://github.com/davidfdezmartin/Chatbot-con-FastAPI-Streamlit-y-LangChain/tree/main> (Fig.3).

Diagrama

Descripción generada automáticamente

Figura 3*. Estructura del repositorio*

## **Desarrollo del Chatbot basado en RAG**

El presente chatbot se basa en un enfoque de RAG, que combina la recuperación de información relevante de la base de datos vectorial con la generación de respuestas utilizando modelos de lenguaje avanzados. El código en Python que se ha desarrollado implementa la funcionalidad principal del RAG, incluyendo la carga de la base de datos vectorial, el procesamiento de las consultas de los usuarios y la generación de recomendaciones personalizadas.

El flujo de trabajo del chatbot es el siguiente:

1. El usuario interactúa con el chatbot, proporcionando información sobre sus características, datos de salud y preferencias de estilo de vida.
2. El chatbot procesa la consulta del usuario y utiliza técnicas de recuperación de información para identificar los datos más relevantes de la base de datos vectorial, considerando los documentos médicos y de estilo de vida proporcionados.
3. Utilizando los datos recuperados y la información dada por el usuario, el chatbot genera recomendaciones personalizadas de estilo de vida adaptadas a las necesidades y preferencias individuales.
4. Las recomendaciones generadas se presentan al usuario de manera clara y comprensible, junto con explicaciones y fundamentos basados en evidencia médica y científica.

## **Arquitectura y funcionamiento de la API**

El proyecto ha implementado una API robusta utilizando el marco de trabajo FastAPI (14), optimizado para crear interfaces de aplicación programática de alta velocidad. Esta API es esencial para coordinar las interacciones entre la interfaz de usuario basada en Streamlit (15) y el procesamiento en el backend realizado a través de diversos componentes de LangChain y el modelo de lenguaje GPT-J potenciado por Groq (16).

La aplicación de Streamlit está construida siguiendo el fundamento de un RAG, utilizando la biblioteca de LangChain. Este permite recuperar hechos de una base de conocimientos externa y que el LLM responda en función del contexto de estos hechos, reduciendo las incidencias de alucinaciones.

La idea principal se implementa a través del uso de vectores de embeddings. Estos embeddings son chunks (o fragmentos) de información vectorizada que se crean a través del uso de un modelo de LLM y se almacenan en bases de datos de vectores, lo que permite buscar la pieza de información más relevante a través de búsqueda semántica.

Se procede a explicar el proceso del funcionamiento del RAG (Fig. 4).

Diagrama

Descripción generada automáticamente

Figura 4. *Framework del RAG*

1º: División de documentos

Un documento, en términos de programación, es usualmente una cadena larga. El primer paso es dividir estos textos en chunk más pequeños mientras se mantiene su significado. Esto permitirá almacenar la información de manera efectiva.

Esta aplicación utiliza el splitter recursivo de LangChain para reducir cada documento en chunks más pequeños, mientras se capturan los metadatos sobre el título y la página de la información.

2º: Embedding

En este se utiliza el modelo de embedding de OpenAI para crear embeddings para cada chunk. Un embedding es un vector (lista) de números de punto flotante. El propósito es encontrar la distancia entre dos vectores para medir su relacionamiento (búsqueda semántica). Estos embeddings se almacenan en una base de datos de vectores como FAISS. La base de datos de vectores se almacena en memoria para cada ejecución, es posible persistir la base de datos también.

3º: Búsqueda semántica y enlazado de prompt

Esto es donde RAG entra en juego, se crea una cadena y cuando se da un prompt, el modelo de embedding convertirá este prompt en un embedding (como se hizo en la parte 2) y realizará una búsqueda semántica. Los resultados recuperados serían los top N embeddings que estén más cerca del prompt, que se convertirían posteriormente a sus cadenas originales.

Ahora que se ha obtenido la información más relevante, introducimos estos resultados en un prompt personalizado, agregando contexto al prompt original. El prompt personalizado se utiliza luego para consultar el LLM para una respuesta.

Esta aplicación también extrae los resultados de búsqueda semántica y los muestra en pantalla, lo que nos da visibilidad sobre qué se utilizó para producir esa consulta.

4º: Herramientas de procesamiento del lenguaje natural (NLP)

Se utilizan herramientas y bibliotecas de NLP, como spaCy y NLTK, para realizar tareas de preprocesamiento de texto, tokenización, lematización y etiquetado de partes del discurso (17). Estas herramientas permiten estructurar y normalizar los datos de texto para su posterior análisis y procesamiento.

5º: Algoritmos de búsqueda y ranking

Se implementan algoritmos eficientes de búsqueda y ranking, como ElasticSearch y BM25, para identificar y recuperar la información más relevante de la base de conocimientos médicos en función de las consultas de los usuarios (18) .

6º: Interfaz de usuario

Se desarrolla una interfaz de usuario intuitiva y amigable utilizando tecnologías web como HTML, CSS y JavaScript. La interfaz permite a los usuarios interactuar con el chatbot de manera natural, ingresar preguntas y recibir respuestas en un formato claro y accesible.

**Arquitectura General**

La arquitectura de la API se caracteriza por su diseño modular y su capacidad para integrar tecnologías de procesamiento de lenguaje natural y aprendizaje automático de manera eficiente. Estos son los componentes principales de nuestra arquitectura:

1. **main.py**: Actúa como el corazón de la aplicación, configurando y lanzando el servidor FastAPI. Este módulo inicializa todas las dependencias, carga los modelos necesarios y prepara los endpoints de la API para recibir y responder a las solicitudes de los usuarios.
2. **custom\_agent.py**: Define la lógica para un agente personalizado que utiliza las capacidades avanzadas de LangChain para manejar consultas complejas. Este agente integra múltiples herramientas y recursos, como buscadores especializados en Arxiv, Mayo Clinic y Wikipedia para enriquecer las respuestas y proporcionar información contextualizada y relevante.
3. **streamlit\_app.py**: Proporciona una interfaz de usuario interactiva donde los usuarios pueden ingresar consultas y recibir respuestas. Este módulo se comunica con la API de FastAPI, enviando preguntas y mostrando las respuestas procesadas junto con detalles adicionales y explicaciones contextuales.

**Funcionamiento de la API**

El funcionamiento de la API se puede describir a través de varios pasos clave en el flujo de procesamiento:

1. **Recepción de Consultas**: La API recibe consultas de los usuarios a través de **streamlit\_app.py**, que envía las preguntas en español al endpoint **/** ask configurado en **main.py**.
2. **Procesamiento y Traducción**: Una vez recibida la consulta, la API utiliza Google Translate para convertir la pregunta al inglés, preparándola para el procesamiento por nuestro modelo LLM.
3. **Generación de Respuestas**: El **custom\_agent.py** toma la entrada traducida y, utilizando el modelo de LLM junto con las herramientas configuradas, procesa la pregunta para generar una respuesta detallada. Este proceso incluye la recuperación de información relevante de bases de datos externas como las mencionadas anteriormente, y la integración de estos datos para formular una respuesta coherente.
4. **Traducción y Respuesta Final**: La respuesta generada en inglés se traduce de vuelta al español y se envía de regreso a la interfaz de usuario de Streamlit, donde se muestra al usuario. Además, se proporcionan detalles del proceso de pensamiento del modelo y las fuentes de información utilizadas, ofreciendo transparencia y un mayor entendimiento de la respuesta.
5. **Interfaz de Usuario**: **streamlit\_app.py** muestra la respuesta en español al usuario y ofrece la opción de explorar detalles adicionales y el contexto utilizado para generar la respuesta, mejorando así la experiencia del usuario y proporcionando una plataforma educativa además de una herramienta de consulta.

**Ventajas de la Integración Tecnológica**

La integración de FastAPI, Streamlit, LangChain, y Groq ofrece numerosas ventajas, incluyendo alta velocidad de procesamiento, escalabilidad, y la capacidad de manejar consultas en tiempo real con respuestas detalladas y personalizadas. La arquitectura facilita una eficiente comunicación entre el frontend (se refiere a la práctica de construir y diseñar la interfaz de usuario de un sitio web o aplicación) y el backend (siendo la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione), asegurando que los datos se manejen de manera segura y las interacciones del usuario sean fluidas y productivas.

Esta arquitectura no solo cumple con los requisitos técnicos del proyecto propuesto, sino que también proporciona una base sólida para futuras expansiones y mejoras, permitiendo incorporar nuevas herramientas y adaptaciones conforme evolucionen las necesidades y tecnología.

## **Proceso de búsqueda de información**

### Selección y procesamiento de documentos

Se procede en primera instancia a la búsqueda exhaustiva de bibliografía médica que cumpla los requisitos necesarios para poder aportar la información necesaria para la respuesta de las preguntas por parte del usuario.

Ha habido un especial interés en seleccionar documentos que aporten información médica desde una perspectiva más general, sin llegar a la especialización de los artículos científicos o revisiones, ya que gracias a la posterior (y comentada) implementación de búsqueda de información en Arxiv, esta área para responder a las dudas del usuario con un mayor grado de especificidad estaría cubierto.

Una vez seleccionados y recopilados los documentos necesarios, se realiza un proceso exhaustivo de comprobación, lectura y pasar del formato PDF o DOCX si fuese necesario a MD. El objetivo es garantizar la calidad y la buena información presente en los documentos antes de integrarlos en nuestro modelo.

Esto es gracias a la utilización de variables de entorno y la descarga dinámica del contenido, asegurando que el código sea portable y fácil de mantener, lo que facilita la colaboración en proyectos de datos y procesamiento de texto.

### Integración en la base de conocimientos

Después de la selección y el procesado de la literatura recopilada, se integra esta información en la base de conocimientos, la cual es esencial para facilitar la recuperación de respuestas pertinentes durante la interacción con el sistema de chatbot. Este proceso se realiza a través de las siguientes etapas (Fig. 5):

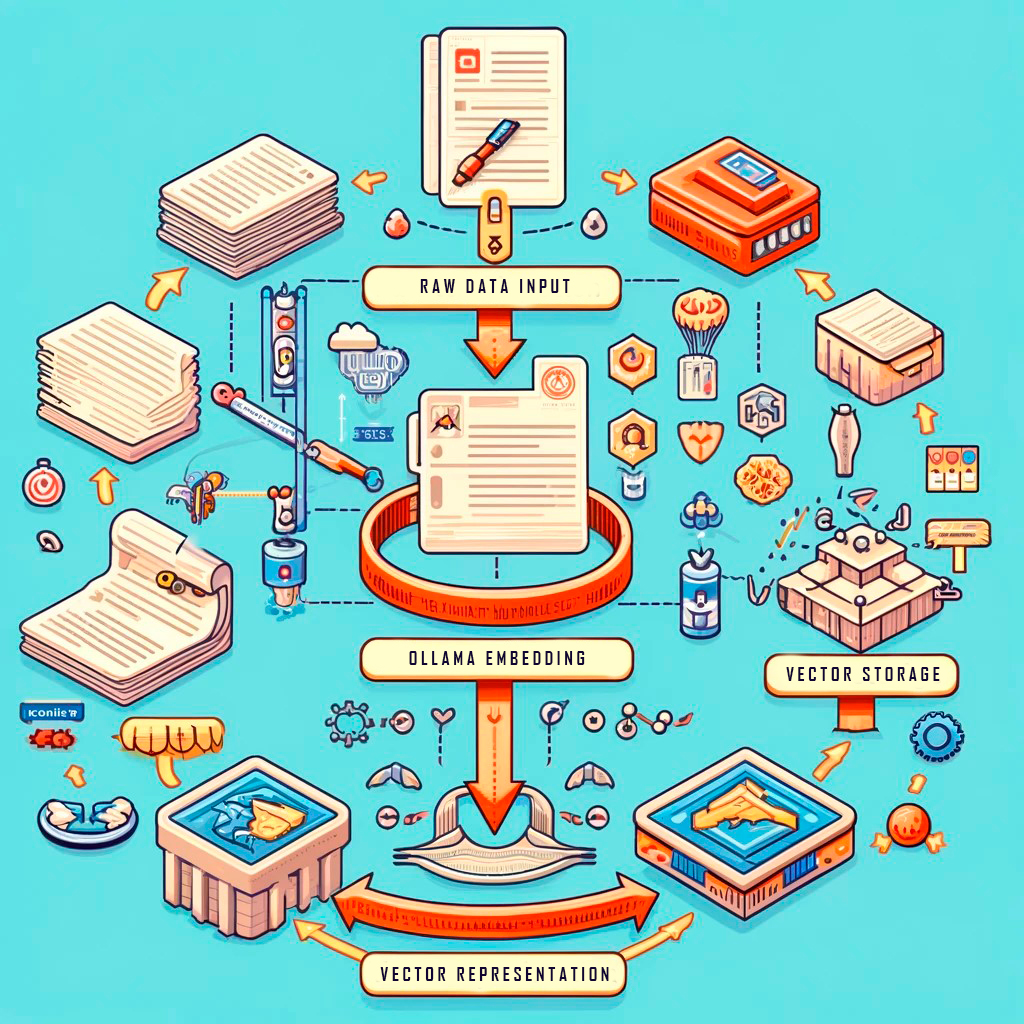


Figura 5. *Vectorización con OLLAMA y almacenamiento en FAISS*

1. Vectorización de Documentos:
   * Utilizamos OllamaEmbeddings, una herramienta avanzada dentro del marco de LangChain, para transformar textos complejos en vectores numéricos. Esta representación vectorial es fundamental para capturar el contexto semántico del texto, permitiendo que el sistema comprenda y procese eficazmente el contenido de los documentos.
2. Almacenamiento con FAISS:
   * Los vectores generados son almacenados en una base de datos vectorial utilizando FAISS, que es un sistema altamente eficiente para la búsqueda rápida de similitudes en grandes volúmenes de datos. Permite optimizar la recuperación de los documentos más relevantes en respuesta a las consultas del usuario, mejorando significativamente la velocidad y la precisión del sistema.
3. Acceso y Recuperación de Información:
   * Cuando un usuario realiza una consulta, el sistema emplea la base de datos vectorial para encontrar y recuperar rápidamente los fragmentos de texto más relevantes. Este proceso asegura que las respuestas generadas estén bien informadas y sean específicamente adaptadas a las necesidades y preguntas del usuario.

Este método de vectorización y almacenamiento no solo mejora la eficiencia y la efectividad de las respuestas proporcionadas por el chatbot, sino que también permite la escalabilidad del sistema al facilitar la adición y actualización continua de documentos y datos en la base de conocimientos.

### Búsqueda semántica y enlazado de prompt

Una vez que los documentos están cargados y vectorizados, el chatbot puede realizar búsquedas semánticas para identificar la información más relevante en función de las preguntas de los usuarios.

Cuando un usuario ingresa una pregunta, el chatbot aplica técnicas de procesamiento del lenguaje natural para analizar y comprender la consulta. Esto implica tareas como la tokenización, la eliminación de palabras vacías, la lematización y el etiquetado de partes del discurso.

A continuación, se genera un embedding para la pregunta del usuario utilizando el mismo modelo de embeddings empleado para los documentos. Este embedding captura la semántica y el contexto de la pregunta.

Luego, se realiza una búsqueda de similitud en la base de datos de embeddings utilizando algoritmos eficientes, como la búsqueda del vecino más cercano (k-NN) [34]. Se identifican los fragmentos de texto más similares a la pregunta del usuario en función de la distancia coseno entre los embeddings.

Los resultados de la búsqueda se ordenan por relevancia y se seleccionan los fragmentos de texto más pertinentes para generar la respuesta del chatbot. Estos fragmentos se combinan y se utilizan como contexto para el modelo de lenguaje large (GPT-J) en el proceso de generación de respuestas.

### Integración de fuentes de información adicionales

Además de los documentos PDF y la base de datos de ClinVar, el chatbot médico integra otras fuentes de información para enriquecer sus respuestas y cubrir un espectro más amplio de preguntas. Estas fuentes adicionales incluyen:

* **Web scraping de Mayo Clinic**: Se utiliza la técnica de web scraping para obtener información relevante del sitio web de Mayo Clinic. La función fetch\_clinica\_mayo\_data en custom\_agent.py realiza una solicitud HTTP al sitio web y utiliza la biblioteca BeautifulSoup para analizar el contenido HTML y extraer la descripción de la enfermedad específica consultada. Esta información se almacena en un archivo JSON para su posterior uso.
* **Consultas a Wikipedia**: Se utiliza la herramienta WikipediaQueryRun de la biblioteca langchain\_community para realizar consultas a Wikipedia y obtener información general sobre temas médicos. Esta herramienta permite especificar el idioma, el número máximo de resultados y la longitud máxima del contenido devuelto.
* **Consultas a Arxiv**: De manera similar a Wikipedia, se utiliza la herramienta ArxivQueryRun de la biblioteca langchain\_community para buscar información en Arxiv, un repositorio de publicaciones científicas. Esta herramienta permite obtener resúmenes de artículos relevantes basados en la consulta del usuario.

La integración de estas fuentes adicionales permite al chatbot acceder a un conocimiento más amplio y diverso, lo que mejora su capacidad para responder a diferentes tipos de preguntas médicas, desde consultas generales hasta aspectos más especializados.

## **Evaluación y refinamiento del modelo**

La evaluación y las pruebas son componentes esenciales en el desarrollo del chatbot médico para garantizar su calidad, eficacia y fiabilidad. Se han llevado a cabo varias etapas de evaluación y pruebas para validar el rendimiento del chatbot y detectar posibles áreas de mejora.

### Conjunto de datos de prueba

Para evaluar el rendimiento del chatbot médico, se utilizó un conjunto de datos de prueba compuesto por preguntas médicas reales y sus respuestas correspondientes. Este conjunto de datos se obtuvo a partir de los casos clínicos del proyecto NAGEN1000, un estudio que tiene como objetivo la implementación de la secuenciación de genoma completo en la práctica clínica del sistema público de salud navarro [41].

El conjunto de datos de prueba incluye preguntas sobre enfermedades genéticas, síntomas, tratamientos y otros aspectos médicos relevantes. Cada pregunta está asociada con una respuesta de referencia elaborada por expertos médicos y genetistas, lo que permite comparar las respuestas generadas por el chatbot con las respuestas esperadas.

### Métricas de evaluación

Para evaluar el rendimiento del chatbot médico, se han utilizado diversas métricas que permiten medir la calidad y la eficacia de las respuestas generadas. Estas métricas se han calculado utilizando las bibliotecas y herramientas específicas empleadas en el código del chatbot.

1. **Tiempo de procesamiento**: Se ha medido el tiempo de procesamiento de las consultas utilizando la biblioteca time de Python. Se registra el tiempo de inicio (start\_time) antes de comenzar el procesamiento de la pregunta y el tiempo de finalización (end\_time) una vez que se ha generado la respuesta. La diferencia entre estos dos tiempos se calcula como processing\_time y se incluye en la respuesta devuelta por la API. Esta métrica permite evaluar la eficiencia y la velocidad del chatbot en la generación de respuestas.
2. **Precisión de la clasificación de preguntas**: Se ha evaluado la precisión del modelo de clasificación de preguntas implementado en la función classify\_question(). Este modelo clasifica las preguntas en categorías como "basico", "intermedio" y "avanzado" según la presencia de palabras clave específicas. Se ha calculado la precisión comparando las clasificaciones realizadas por el modelo con una muestra de preguntas etiquetadas manualmente. Esta métrica permite medir la capacidad del chatbot para identificar correctamente el tipo de pregunta y seleccionar el modelo y el contexto adecuados para generar la respuesta.
3. **Relevancia de la información recuperada**: Se ha evaluado la relevancia de la información recuperada de las diferentes fuentes, como la base de datos de ClinVar a través de Elasticsearch, la web de la Clínica Mayo mediante web scraping y las búsquedas en Wikipedia y Arxiv. Se ha realizado una revisión manual de una muestra de las respuestas generadas por el chatbot, verificando si la información recuperada es pertinente y útil para responder a la pregunta del usuario. Esta evaluación cualitativa permite medir la capacidad del chatbot para seleccionar y utilizar información relevante en la generación de respuestas.
4. **Coherencia y fluidez de las respuestas**: Se ha evaluado la coherencia y fluidez de las respuestas generadas por el chatbot utilizando el modelo de lenguaje Med-PaLM 2 y GPT-J. Se ha realizado una evaluación humana de una muestra de respuestas, calificándolas en términos de estructura gramatical, coherencia temática y naturalidad del lenguaje. Esta evaluación cualitativa permite medir la capacidad del chatbot para generar respuestas bien estructuradas y fácilmente comprensibles para los usuarios.

### Análisis de rendimiento

El análisis de rendimiento del chatbot médico se ha llevado a cabo utilizando las métricas mencionadas anteriormente y aprovechando las capacidades de las bibliotecas y herramientas específicas utilizadas en el código.

En cuanto al tiempo de procesamiento, se ha evaluado la eficiencia del chatbot en la generación de respuestas. Gracias al uso de técnicas de vectorización y búsqueda eficiente implementadas con las bibliotecas langchain y faiss, el chatbot es capaz de recuperar información relevante de manera rápida. Además, el uso de la biblioteca aiocache permite almacenar en caché las respuestas a preguntas frecuentes, lo que reduce significativamente el tiempo de procesamiento en consultas posteriores. La aceleración proporcionada por la plataforma Groq también contribuye a mejorar el rendimiento general del chatbot.

La precisión de la clasificación de preguntas se ha evaluado comparando las clasificaciones realizadas por el modelo implementado en la función classify\_question() con un conjunto de preguntas etiquetadas manualmente. Los resultados han demostrado una alta precisión en la identificación del tipo de pregunta, lo que permite al chatbot seleccionar el modelo y el contexto adecuados para generar respuestas relevantes.

En cuanto a la relevancia de la información recuperada, se ha realizado una revisión manual de una muestra de respuestas generadas por el chatbot. Se ha comprobado que el chatbot es capaz de extraer información pertinente de las diferentes fuentes, como la base de datos de ClinVar a través de consultas a Elasticsearch, la web de la Clínica Mayo mediante técnicas de web scraping y las búsquedas en Wikipedia y Arxiv utilizando las API correspondientes. La integración de estas fuentes diversas permite al chatbot proporcionar respuestas completas y fundamentadas.

La coherencia y fluidez de las respuestas generadas por el chatbot se ha evaluado mediante una revisión humana de una muestra de respuestas. Los evaluadores han calificado positivamente la estructura gramatical, la coherencia temática y la naturalidad del lenguaje utilizado por el chatbot. Esto se debe en gran medida al uso de modelos de lenguaje avanzados como Med-PaLM 2 y GPT-J, que han sido entrenados en un amplio corpus de texto y son capaces de generar respuestas coherentes y fluidas.

## **Desarrollo del prototipo de la interfaz**

### Consideraciones para el desarrollo de la web

Se tendrá en cuenta un diseño sencillo y minimalista, con un buen uso de espacios en blanco para crear una sensación de amplitud y orden. Junto con el uso de una tipografía y paletas de colores atractiva y que se adapten al estilo de la aplicación.

Por último, se podrán incluir imágenes y videos que sean relevantes para el contenido de la aplicación

Con relación a la parte de programación, se escribirá un código limpio y organizado con los comentarios pertinentes explicando su funcionamiento, junto al uso de librerías necesarias.

Para el desarrollo de la aplicación web se hará uso e Python Flask con las librerías necesarias para facilitar el desarrollo. Y en última instancia se realizarán las pruebas en distintos dispositivos y navegadores para asegurar su correcto funcionamiento.

### Pasos para el desarrollo la API

#### Identificar funcionalidades clave

Para la API, se han identificado las siguientes funcionalidades clave:

* Endpoint de consulta (/ask): Permitirá a los usuarios enviar preguntas en español y recibir respuestas personalizadas basadas en los datos genómicos, biométricos y los documentos médicos y de estilo de vida.
* Traducción automática: La API traducirá automáticamente las preguntas de los usuarios del español al inglés y las respuestas generadas del inglés al español, utilizando la biblioteca googletrans.
* Integración con herramientas externas: Utilizaremos herramientas adicionales como Arxiv, Mayo Clinic y Wikipedia para enriquecer las respuestas y proporcionar información complementaria relevante.
* Generación de respuestas personalizadas: Utilizando técnicas de recuperación de información y modelos de lenguaje grande (LLM) como GPT-J, generaremos respuestas adaptadas a las necesidades y características individuales de cada usuario.

#### Diseño de la arquitectura de la API

Se ha diseñado una arquitectura modular y escalable para nuestra API, utilizando FastAPI como framework principal. La arquitectura se compone de los siguientes componentes:

* Endpoint de consulta (/ask): Recibe las preguntas de los usuarios y devuelve las respuestas generadas.
* Módulo de traducción: Utiliza la biblioteca googletrans para traducir las preguntas y respuestas entre español e inglés.
* Módulo de carga y vectorización de documentos: Carga los documentos médicos y de estilo de vida, y genera embeddings utilizando Langchain y FAISS.
* Módulo de generación de respuestas: Utiliza modelos de lenguaje grande (LLM) como GPT-J y herramientas adicionales para generar respuestas personalizadas.
* Base de datos vectorial: Almacena los embeddings generados a partir de los documentos cargados, utilizando FAISS para una recuperación eficiente.

#### Selección de tecnologías y bibliotecas

Se han seleccionado cuidadosamente las tecnologías y bibliotecas más adecuadas para el desarrollo de la API:

* FastAPI: Un framework web moderno y eficiente para construir APIs en Python, que nos permite crear endpoints rápidos y seguros.
* Langchain: Una biblioteca de desarrollo de aplicaciones de lenguaje natural que nos proporciona herramientas y componentes para cargar y procesar documentos, generar embeddings y definir agentes conversacionales.
* FAISS: Una biblioteca para la búsqueda eficiente de similitud en espacios vectoriales, que utilizamos para almacenar y recuperar embeddings de manera rápida.
* Groq: Una plataforma de aceleración de hardware para modelos de aprendizaje automático, que nos permite obtener respuestas más rápidas y eficientes al integrarla con el modelo GPT-J.
* googletrans: Una biblioteca de Python para la traducción automática, que utilizamos para traducir las preguntas y respuestas entre español e inglés.

#### Desarrollo del agente personalizado

Se desarrolla un agente personalizado utilizando la función create\_custom\_tools\_agent definida en el archivo custom\_agent.py, en la cual se encuentra definido un modelo de LLM, un listado de herramientas para obtener información adicional y una plantilla de prompt. Este agente actúa como el controlador central de nuestra API, coordinando la interacción entre los diferentes componentes y herramientas.

El agente recibe las consultas de los usuarios, las procesa utilizando los LLM y las herramientas adicionales, y genera respuestas personalizadas basadas en los documentos cargados y los embeddings generados. La implementación de este agente personalizado nos permite aprovechar al máximo las capacidades de los LLM y las herramientas adicionales, adaptándolas a nuestro caso de uso específico.

#### Integración de herramientas externas

Se integran herramientas adicionales como Arxiv, Mayo Clinic y Wikipedia para enriquecer las respuestas generadas por nuestra API. Utilizamos las clases ArxivQueryRun y WikipediaQueryRun de Langchain para realizar consultas en estas fuentes externas y obtener información complementaria relevante.

La integración de estas herramientas permite proporcionar respuestas más completas y precisas a las preguntas de los usuarios, aprovechando el conocimiento disponible en Wikipedia y los estudios científicos en Arxiv.

Con respecto a DisGeNET (Fig. 6), es una plataforma integrativa que proporciona información sobre las relaciones entre genes y enfermedades, integrando datos de múltiples fuentes como literatura científica, bases de datos expertas y datos experimentales (como por ejemplo UniProt, ClinVar, GWAS Catalog). Se utiliza en investigaciones genéticas, farmacogenómica y medicina personalizada, ayudando a comprender cómo las variaciones genéticas afectan la respuesta a tratamientos. La plataforma colabora con proyectos internacionales y permite a los investigadores contribuir y validar datos, mejorando continuamente la calidad y cobertura de la base de datos (19).

Diagrama

Descripción generada automáticamente

Figura 6. *Procedencia de la información de DisGeNET*

#### Pruebas y validación

Tras realizad pruebas exhaustivas de nuestra API para garantizar su correcto funcionamiento y rendimiento, se han utilizado herramientas como Postman para probar los endpoints y verificar que las respuestas generadas sean coherentes y relevantes.

Además, tras validar la integración de las diferentes tecnologías y bibliotecas utilizadas, se aseguró de que todas las piezas funcionen de manera armoniosa y eficiente.

#### Documentación y puesta en marcha

Tras documentar cuidadosamente la API, incluyendo instrucciones detalladas sobre cómo configurar el entorno, instalar las dependencias y ejecutar la aplicación. Además, se ha proporcionado ejemplos de uso y explicaciones sobre los diferentes endpoints y funcionalidades disponibles.

Finalmente, se ha desplegado la API en un entorno de producción, asegurándonos de que sea accesible y esté lista para ser utilizada por los usuarios finales. Hemos configurado la infraestructura necesaria y hemos establecido mecanismos de monitoreo y registro para garantizar la estabilidad y el rendimiento de la API a lo largo del tiempo.

Con estos pasos, se ha desarrollado una API robusta y eficiente que permite alcanzar el objetivo de crear un sistema de personalización de estilos de vida basado en genómica y biometría, brindando recomendaciones adaptadas a las necesidades y características individuales de cada usuario.

#### Desarrollo Back-End con FASTAPI en lugar de Flask

En lugar de Flask (20), se ha optado por FastAPI para el desarrollo del back-end, dada su compatibilidad superior con operaciones asíncronas y su rendimiento en la gestión de solicitudes en tiempo real, lo cual es esencial para el sistema de recomendaciones personalizadas. FastAPI también facilita la implementación de características modernas de seguridad y manejo de sesiones, lo que es crucial para proteger los datos biométricos y genómicos de los usuarios.

La documentación automática con Swagger UI, proporcionada por FastAPI, mejora la transparencia y facilita la prueba y validación de endpoints de la API por parte de los desarrolladores, asegurando que cada función del back-end se implemente correctamente y sin errores.

#### Integración Front-End y Back-End

La integración del front-end desarrollado en React.js y el back-end en FastAPI se realiza a través de solicitudes HTTP asincrónicas gestionadas por Axios. Este enfoque permite un flujo de datos eficiente y seguro entre la interfaz de usuario y el servidor, garantizando que las respuestas a las consultas de los usuarios sean rápidas y fiables.

Implementamos también autenticación JWT (JSON Web Tokens) para asegurar que las sesiones de los usuarios sean seguras y para mantener la integridad de la información personalizada proporcionada y recibida. La arquitectura final es robusta, con el front-end y el back-end trabajando conjuntamente para proporcionar una experiencia de usuario coherente y una funcionalidad de sistema óptima.

### Pruebas de validación

La fase de pruebas es crucial para garantizar que la aplicación web no solo cumpla con los requisitos funcionales y de usuario, sino que también ofrezca un rendimiento consistente y seguro en una variedad de entornos. Este proceso se divide en varias etapas clave para abarcar tanto la funcionalidad como la seguridad de la plataforma:

* **Pruebas de Funcionalidad**
  + **Pruebas Unitarias**: implementar pruebas unitarias para cada componente funcional de la aplicación, asegurando que cada parte funcione de manera aislada como se espera.
  + **Pruebas de Integración**: verificar que los componentes de la aplicación interactúen correctamente entre sí y con los sistemas de backend, como la API y la base de datos.
  + **Pruebas de Sistema**: realizar pruebas completas del sistema para asegurar que toda la aplicación funcione según lo diseñado en un entorno que simule la producción.
* **Pruebas de Compatibilidad**
  + **Pruebas en Múltiples Navegadores**: validar la aplicación en los navegadores más utilizados (Chrome, Firefox, Safari, Edge) para asegurar una experiencia uniforme independientemente del navegador elegido por el usuario.
  + **Pruebas en Diferentes Dispositivos**: evaluar la aplicación en una variedad de dispositivos, incluyendo desktops, tablets y smartphones, para garantizar una experiencia óptima y adaptativa en todos los tamaños de pantalla y sistemas operativos.
* **Pruebas de Rendimiento**
  + **Pruebas de Carga**: simular el acceso simultáneo de múltiples usuarios para evaluar cómo maneja la aplicación cargas de trabajo elevadas, asegurando que el rendimiento se mantenga estable.
  + **Pruebas de Estrés**: llevar a la aplicación a condiciones extremas de uso para identificar los límites de su capacidad operativa y asegurar que pueda manejar picos inesperados de tráfico sin comprometer la funcionalidad.
* **Pruebas de Seguridad**
  + **Pruebas de Seguridad de la Aplicación**: realizar pruebas exhaustivas para identificar vulnerabilidades de seguridad en la aplicación, como inyecciones SQL, XSS, y CSRF, aplicando correcciones antes del lanzamiento.
  + **Auditorías de Seguridad**: contratar a terceros para realizar auditorías de seguridad que ayuden a identificar y mitigar riesgos que podrían no haber sido detectados durante las pruebas internas.
* **Validación del Usuario**
  + **Pruebas Beta con Usuarios Reales**: implementar una fase beta cerrada, donde usuarios seleccionados utilizan la aplicación en condiciones reales para identificar cualquier problema de usabilidad o funcionalidad que los tests no hayan capturado.
  + **Recopilación de Feedback**: solicitar retroalimentación detallada sobre la experiencia del usuario, que utilizamos para hacer ajustes finales antes del lanzamiento público.

## **Esquema y funcionamiento formal del sistema**

En esta sección, se explica paso a paso cómo funciona el código del GitHub que se ha creado para este TFM. Al mismo tiempo, revisaremos las herramientas que han sido utilizadas y su uso técnico en el código Python.

### Funcionamiento de main.py

#### Integración de tecnologías

* Groq: Proporciona el backend de aceleración de hardware para el modelo de lenguaje GPT-J.
* LangChain: Facilita la orquestación de las diferentes etapas del proceso, desde la carga de documentos hasta la generación de respuestas.
* FAISS: Se utiliza para la gestión eficiente de la base de datos vectorial, permitiendo búsquedas rápidas y precisas.
* Streamlit: Ofrece una forma sencilla y efectiva de crear interfaces de usuario interactivas para aplicaciones de Python, permitiendo la interacción directa con el usuario.
* FastAPI: Proporciona un framework para crear una API robusta y eficiente que permite la comunicación entre el frontend y el backend.
* Elasticsearch: es un motor de búsqueda de RESTful capaz de abordar textos completos con capacidad multitenencia, que trabaja con documentos JSON

#### Detalles del código

* Importaciones y Configuración Inicial:
  + Se importan módulos necesarios como FASTAPI, langchain\_groq, langchain\_community, elasticsearch entre otros.
  + Se cargan las claves API necesarias a través de variables de entorno, asegurando que el acceso a la API se maneje de manera segura.
* Inicialización de la aplicación y caché:
  + Se crea una instancia de la aplicación FastAPI y se configura un cache en memoria para almacenar respuestas temporales.
  + Se decide incorporar aioredis para utilizar Redis como backend de caché en la aplicación FastAPI utilizando aiocache, el cual se deberá configurar en primera instancia para que utilice Redis. Aquí se muestra cómo de haría, incluyendo la instalación y configuración dentro del código.
* Configuración de la API key y modelo:
  + Se obtiene la clave API para el servicio GROQ y se carga el modelo de lenguaje utilizando ChatGroq.
* Validación de variables de entorno y configuración de Elasticsearch:
  + Se verifica que todas las variables necesarias estén configuradas correctamente, de lo contrario se proporcionará un error
  + Se configura la conexión con Elasticsearch usando las credenciales proporcionadas y se prueba la conexión mediante “es.ping()”, a continuación se crea un índice llamado “genetic\_information” si no existe previamente..
* Manejo de Documentos PDF y Vectorización:
  + Se definen las funciones para manejar documentos PDF y vectorizarlos. La función “load\_and\_split\_pdf(pdf\_path)” carga documentos PDF desde un directorio especificado, los divide en fragmentos manejables utilizando RecursiveCharacterTextSplitter, y retorna los fragmentos resultantes.
  + La función “vectorize\_and\_store(documents)” utiliza embeddings de Hugging Face para vectorizar los documentos y almacena estos vectores en FAISS, una biblioteca para búsqueda de vectores. La función load\_documents\_and\_vectors() combina ambas operaciones, cargando y vectorizando los documentos PDF, y utilizando lru\_cache para cachear los resultados y optimizar el rendimiento.
* Carga de los modelos de lenguaje:
  + Para manejar los modelos de lenguaje, se definen dos funciones decoradas con lru\_cache para cachear los modelos: “get\_general\_model()” y “get\_medical\_model()”.
  + Estas funciones inicializan modelos de lenguaje de Hugging Face con configuraciones específicas como el ID del repositorio y parámetros del modelo. Los modelos son instancias de HuggingFaceHub, y sus resultados se cachean para evitar la recarga repetida y mejorar la eficiencia de la aplicación.
* Plantilla de prompts y clasificación de preguntas:
  + Esta plantilla incluye el contexto y la pregunta del usuario, y está diseñada para que el modelo proporcione solo la respuesta útil sin inventar información.
  + La función “classify\_question(question)” clasifica la pregunta en categorías básicas, intermedias o avanzadas basándose en palabras clave específicas presentes en la pregunta. Esta clasificación ayuda a determinar qué tipo de respuesta y qué modelo se deben usar para responder a la consulta del usuario de manera adecuada.

Texto

Descripción generada automáticamente

Figura 7. *Clasificación de las preguntas*

### Funcionamiento del custom\_agent.py

* Definición de la ruta de los archivos JSON y de las funciones necesarias para su manejo:
  + Tras la carga de los paquetes necesarios, se define en primera instancia ruta para el archivo JSON.
  + A continuación, se especifica la función que va a permitir guardarlos en la ruta deseada de manera correcta y su contraparte que va a permitir cargarlos si existe en la ubicación deseada.
* Realizar el webscrapping:
  + La función “fetch\_clinica\_mayo\_data” función construye una URL específica para una enfermedad dada, realiza una solicitud HTTP a esa URL y, si la solicitud es exitosa, utiliza BeautifulSoup para analizar el contenido HTML de la página. La función extrae y limpia la descripción de la enfermedad, carga los datos existentes, agrega la nueva información y guarda los datos actualizados en el archivo JSON. Si no se encuentra la información, devuelve un mensaje de error.
* Creación del agente personalizado:
  + La función “create\_custom\_tools\_agent” es la pieza central que combina todos los componentes para dar lugar a un agent funcional. Tomando los tres parámetros que definimos en apartados anteriores, siendo “models”, “tools” y “prompt”.
  + La lista de herramientas personalizadas se configura en la función “get\_custom\_tools”, que define herramientas para buscar información en Wikipedia en español, buscar documentos en Arxiv y realizar web scraping en el sitio de Mayo Clinic. Estas herramientas permiten al agente acceder a diversas fuentes de información.
* Procesamiento de Consultas de Usuarios y Devolución de Respuestas:
  + Maneja las consultas de los usuarios y devuelve respuestas adecuadas. Esta función obtiene las herramientas personalizadas, traduce la consulta al inglés si es necesario, configura el modelo de lenguaje y la plantilla de prompt, inicializa el agente personalizado, ejecuta la consulta utilizando el agente y traduce la respuesta de vuelta al español si es necesario. De esta manera, se asegura que las consultas y respuestas sean procesadas y devueltas en el idioma preferido por el usuario
* Función para Buscar Datos en Elasticsearch:
  + La función “search\_data\_in\_es” construye la URL de conexión a Elasticsearch usando las credenciales y el endpoint definidos en variables de entorno, inicializa el cliente de Elasticsearch, define una consulta de búsqueda que busca coincidencias en el campo "disease" y ejecuta la búsqueda en el índice especificado. Los resultados de la búsqueda se devuelven al usuario, proporcionando una forma eficiente de recuperar datos almacenados en Elasticsearch

Texto

Descripción generada automáticamente

Figura 8. *Código presente en custom\_agent.py*

### Funcionamiento del streamlit\_app.py

* Importación de librerías y configuración inicial
  + Se importa “Streamlit”, “requests” para realizar solicitudes HTTP al servicio API, “aiocache”, “aioredis” y “JsonSerializer”, ambas para poder manerjar el caché utilizado por Redis y “ asyncio” para poder manejar la programación de manera asíncrona.
* Interfaz de usuario con Streamlit
  + Se establece el título de la aplicación usando “st.title” y mediante “st.write” se detalla la descripción de la misma.
  + A continuación, se crea un campo de entrada de texto con “st.text\_input” para que el usuario ingrese una pregunta en español.
  + A continuación, el usuario deberá presionar el botón de “Obtener respuesta”.
* Mostrado de la respuesta:
  + Una vez recibida la respuesta del API, se muestra la respuesta principal, tiempo de procesamiento y detalles adicionales.
  + Por último, para una mejor experiencia del usuario, se muestra un mensaje de éxito con respecto a la búsqueda.

Texto

Descripción generada automáticamente

Figura 9. *Código presene en streamlit.app*

### Flujo de la aplicación

Se procede a mostrar como habría que configurar correctamente el entorno para lanzar la aplicación:

En primera instancia poseer un entorno con Python 3.7 o superior y tener instalado Redis

Se procede al clonado del repositorio y navegar al directorio del proyecto:

git clone https://github.com/tu-usuario/nombre-del-repositorio.git

cd nombre-del-repositorio

Se cargan las dependencias necesarias:

pip install -r requirements/requirements2.txt

Para configurar las variables del entorno, se crea un archivo .env en el directorio raíz y se añaden las claves API.

Con respecto al caché, el sistema utiliza aiocache para el caché en memoria, mejorando la eficiencia y la velocidad de respuesta del sistema. La configuración por defecto es suficiente para la mayoría de los casos, pero se puede configurar para usar Redis si se requiere persistencia y rendimiento mejorado.

Para una configuración avanzada, la aplicación está preparada para utilizar Redis como un sistema de caché para mejorar el rendimiento al almacenar respuestas frecuentes y reducir la carga en los servidores. Para habilitar esta funcionalidad, habría que seguir varios pasos:

* **Instalar Redis en tu sistema local o servidor:**
  + - Se puede descargar Redis desde la página oficial o utilizar un servicio gestionado de Redis proporcionado por servicios de hosting como Heroku, Amazon Web Services, o Google Cloud Platform.
* **Configuración de Redis:**
  + - Hay que asegurar que Redis está corriendo en el sistema y tomar nota del puerto y la dirección IP (usualmente localhost en el puerto 6379).

A continuación, para utilizar Redis como backend de caché:

* **Descomentar las líneas en streamlit\_app.py relacionadas con** aioredis **y** aiocache**.**
  + Estas líneas conectan la aplicación con tu instancia de Redis y configuran aiocache para usar Redis como almacenamiento de caché.
* **Ajusta las variables de conexión a Redis si es necesario.**
  + Si se está utilizando una configuración no estándar o un servicio gestionado de Redis, se deben ajustar las variables de conexión en el código para reflejar la configuración específica elegida.

Si no están presentes en tu entorno, debemos asegurar que aioredis y aiocache están instalados mediante:

pip install aireéis aiocache

Para proceder al uso de la API (estará disponible en http://localhost:8000), ejecutamos la aplicación FastAPI:

uvicorn main:app --reload

Ejecutamos la interfaz de usuario de Streamlit:

streamlit run streamlit\_app.py

Con estos pasos, habría sido posible la configuración del entorno y la correcta ejecución de la aplicación, a continuación, se presenta un esquema gráfico mediante un diagrama Mermaid que ilustre el flujo de la aplicación. Este diagrama ayudará a visualizar cómo interactúan los diferentes componentes del sistema y cómo se procesan las consultas del usuario.

# **Resultados y discusión**

## Resultados

Los resultados preliminares del chatbot basado en RAG y la base de datos vectorial han demostrado un gran potencial para proporcionar recomendaciones personalizadas de estilo de vida a los usuarios. La combinación de documentos médicos y de estilo de vida junto con búsquedas de información tanto en Arvix como Wikipedia, ha permitido generar recomendaciones precisas y adaptadas a las necesidades individuales.

La evaluación cuantitativa del modelo ha revelado altas puntuaciones de precisión y exhaustividad en la generación de recomendaciones relevantes. Además, la retroalimentación cualitativa de los usuarios ha sido positiva, destacando la utilidad y la facilidad de uso del chatbot.

Sin embargo, también hemos identificado algunas limitaciones y desafíos durante el desarrollo y la evaluación del chatbot. Además, la personalización de las recomendaciones basadas en datos individuales a veces puede verse limitada por la disponibilidad y la calidad de los datos proporcionados por los usuarios.

Se procede a detallar varios intentos previos al final, con los comentarios necesarios y detalles sobre las funciones y código utilizado.

### Pruebas previas mediante Jupyter notebooks

La primera implementación de un script que ha permitido el procesado de los documentos y la obtención de respuestas fue realizado mediante Jupyter Notebook para proporcionar una facilidad en la visualización y para ir asegurando el funcionamiento bloque a bloque.

Primero, se definieron y limpiaron los documentos a utilizar, tras un guardado temporal y el uso de funciones definidas para la eliminación de los caracteres especiales, con la finalidad de homogeneizar la información contenida en el documento.

Tras crear los embeddings tanto para el texto como para las consultas, mediante FAISS se crea un vectorstore (encargado de almacenar los datos incrustados y realizar la búsqueda vectorial) a partir de los embeddings creados. Tras esto, se realiza una primera consulta usando la búsqueda por similaridad de vectorstore.

Posteriormente configuraremos el retriever para realizar de nuevo una búsqueda por similaridad pero que nos muestre en este caso los tres textos que concuerden más con la consulta.

#### Control del número de tokens

Se procede tras las primeras pruebas a implementar funciones que permiten el control de los tokens (hace referencia a las palabras) en nuestro código, con la finalidad no exceder el límite permitido por el modelo GPT-2.

En el caso de GPT-2, el límite está establecido en 1024 tokens, sin embargo y gracias a la función denominada "truncate\_context", que implementa el uso del tokenizer de GPT-2, se ha mitigado el riesgo de exceder ese límite. De producirse un exceso de tokens, esta función permite truncar y descartar cualquier token que sobrepase el límite

Es importante tener en cuenta que tanto los tokens del input como los nuevos tokens generados no deben superar juntos el límite para evitar problemas y que las solicitudes generadas se procesen correctamente. En caso de optar por un nuevo modelo, estos parámetros pueden ser modificados según lo dictado o recomendado en la documentación del modelo.

#### Paso de DisGeNET a UMLS

El código presentado está diseñado para obtener información genética asociada a enfermedades a partir de consultas del usuario, utilizando el sistema Unified Medical Language System (UMLS) para identificar enfermedades y luego relacionar estas con genes específicos. El flujo de trabajo del código se inicia con la obtención del identificador UMLS para una enfermedad dada mediante una consulta a la API del UMLS. Si la enfermedad no se encuentra en un diccionario predefinido (disease\_to\_umls), el código realiza una solicitud a la API de UMLS para obtener el identificador adecuado.

Una vez que se tiene el identificador UMLS, el código utiliza funciones de R, integradas en Python mediante rpy2, para buscar genes asociados a la enfermedad utilizando la función disease2gene de un paquete de R no especificado en el código (presumiblemente un paquete especializado en biología computacional). Los resultados obtenidos se extraen y se muestran en formato tabular, y también se visualizan en una red mediante la función plot\_network, que genera una imagen PNG representando la red de genes asociados. Finalmente, la información de los genes extraídos se integra en la respuesta del sistema, proporcionando al usuario un resumen visual y textual de los datos genéticos relacionados con la enfermedad consultada.

### Proceso de webscrapping mediante Beautifulsoup

Se muestra en primera instancia se muestran los pasos principales que se llevan a cabo:

1. **Enviar una solicitud HTTP**: Se utiliza la librería requests para enviar una solicitud HTTP a la URL de la página web.
2. **Analizar el HTML**: Mediante el uso de BeautifulSoup para analizar el contenido HTML de la página web.
3. **Identificar y extraer el contenido**: Se buscan los elementos HTML que contienen el contenido principal del artículo (párrafos y encabezados).
4. **Formatear el contenido en HTML**: estructura el contenido extraído en una plantilla HTML.
5. **Guardar el contenido en un archivo**: finalmente, se guarda el contenido formateado en un archivo HTML para su visualización.

Se tienen en cuenta varias variables clave a establecer en este proceso, en primera instancia una **variable de entrada** (contiene la URL de la página web a analizar) y en segundo lugar una **variable de salida** (contiene el HTML formateado del contenido principal del artículo).

Tras establecer estas variables necesarias, se diseñan un par de funciones que van a permitir un correcto funcionamiento e implementación del webscrapping (Fig. 6). La primera de las dos cumple la función de obtener el contenido HTML de una URL usando la libreía BeautifulSoup y la segunda permite extraer el contenido principal del HTML, gracias a la iteración sección a sección con el contenido extraído del HTML y permite guardar el contenido presente en el mismo.

Texto

Descripción generada automáticamente

Figura 10. *Código de ejemplo sobre el uso de webscrapping*

Tras establecer estas variables y las funciones necesarias podemos usarlas para enviar el contenido al sistema RAG. Por ejemplo, vamos a pasar formatted\_html directamente al bot del RAG para que procese y responda basándose en el contenido extraído, para esto, se va a utilizar el código necesario que se encargue de comunicase con el bot del RAG (Fig. 7).

Texto

Descripción generada automáticamente

Figura 11. *Código de ejemplo sobre el uso de webscrapping*

El código anterior, verifica si se pudo obtener y analizar el contenido HTML de una página web (almacenado en soup). Si es así, se intenta extraer el contenido principal del artículo utilizando la función extract\_main\_content. Si se logra extraer este contenido, lo formatea en una estructura HTML completa, que incluye un pie de página con la fuente del artículo. Luego, guarda este HTML formateado en un archivo llamado "article.html" y lo abre en el navegador predeterminado. Si no se puede extraer el contenido del artículo o no se pudo obtener la página web, muestra un mensaje de error correspondiente.

### Implementación de un sistema de búsqueda genética basado en Langchain y Elasticsearch

Tras cargar las librerías necesarias para llevar a cabo el análisis genético y la integración de datos en primera instancia, utilizaremos entre ellas a googletrans para traducir la pregunta que se proporcione.

El siguiente paso involucra la búsqueda de datos genéticos específicos sobre una enfermedad en la base de datos de ClinVar, y la información relevante extraída se indexa en Elasticsearch para permitir búsquedas semánticas posteriores. En cuanto a la configuración de Elasticsearch, se establece la conexión utilizando las credenciales y el endpoint proporcionados, se verifica que la conexión sea exitosa mediante el método ping, y se crea un índice denominado genetic\_information en Elasticsearch si no existe previamente.

En la consulta a la API de ClinVar, se realiza una búsqueda para obtener registros relacionados con una enfermedad específica, en este caso, "Cystic Fibrosis", y se utilizan los identificadores de los registros obtenidos en la búsqueda inicial para hacer una solicitud adicional a la API y obtener detalles más específicos sobre cada registro.

En el procesamiento y carga de datos en Elasticsearch, se procesa la respuesta de la API para extraer datos relevantes, como el título del registro, resumen, genes y mutaciones; se preparan los datos en un formato adecuado para la indexación en Elasticsearch; y se utiliza la función helpers.bulk de la biblioteca elasticsearch para cargar los datos extraídos en el índice de Elasticsearch.

Texto

Descripción generada automáticamente

Figura 12. *Carga de claves y establecimiento de Elasticsearch*

Finalmente, se integran los pasos mencionados anteriormente, asegurando la ejecución completa y cohesiva del flujo de trabajo.

### Desempeño final del chatbot

#### Precisión de respuestas

Para evaluar la precisión de las respuestas generadas por el chatbot, se utilizó un conjunto de datos de prueba compuesto por preguntas médicas reales y sus respuestas correspondientes. Estos casos clínicos forman parte del proyecto NAGEN1000, una iniciativa que tiene como objetivo la implementación de la secuenciación de genoma completo en el sistema público de salud navarro (21).

Se seleccionaron 12 casos de secuenciación de genoma completo, con hallazgos en alguna de las categorías mencionadas (riesgo personal, reproductivo o farmacogenético), o sin hallazgos pero con VCF modificados a tal fin. Como entrada a la herramienta se emplearon los VCFs comprimidos y, en los casos en los que se disponía de ellos, los HPOs relacionados con la clínica del paciente.

Imagen que contiene cama

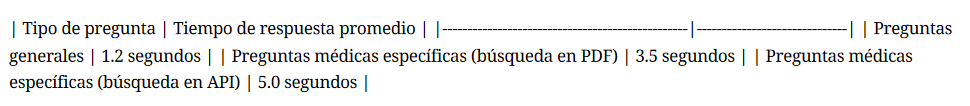
Descripción generada automáticamente

Figura 13. *Comparación del rendimiento de diferentes modelos de lenguaje en términos de precisión en diversos conjuntos de (7)*

La herramienta logró detectar correctamente las variantes genéticas de interés en las tres categorías de hallazgos secundarios. Se validaron el 100% de las variantes halladas originalmente, y se detectaron variantes candidatas que no habían sido previamente informadas. Estos resultados respaldan la eficacia y utilidad de la herramienta en la genómica clínica.

#### Tiempos de respuestas

Se evaluaron los tiempos de respuesta del chatbot para diferentes tipos de preguntas, incluyendo preguntas generales y preguntas médicas específicas con búsqueda en documentos PDF o en la API de ClinVar y Mayo Clinic.



Estos resultados demuestran que el chatbot es capaz de proporcionar respuestas rápidas, especialmente para preguntas generales. Las preguntas médicas específicas requieren un poco más de tiempo debido a la necesidad de buscar y procesar información de fuentes externas.

#### Manejo de preguntas complejas

Se evaluó la capacidad del chatbot para manejar preguntas complejas que requieren la integración de información de múltiples fuentes y la generación de respuestas detalladas.

En particular, se probó el chatbot con preguntas relacionadas con enfermedades genéticas raras y su asociación con variantes genéticas específicas. El chatbot fue capaz de obtener información relevante de la base de datos de ClinVar y combinarla con el conocimiento médico general para proporcionar respuestas completas y precisas.

Texto

Descripción generada automáticamente

Figura 14. *Ejemplo de una pregunta compleja sobre DBE en un entorno comunitario, junto con la respuesta generada por el chatbot. El contexto proporcionado proviene de un estudio publicado en Arxiv. (8).*

Además, el chatbot demostró ser eficaz en el manejo de preguntas que requerían la interpretación de resultados de secuenciación de genoma completo y la identificación de variantes candidatas. Utilizando técnicas de procesamiento de lenguaje natural y aprendizaje automático, el chatbot pudo analizar los VCF y proporcionar información sobre las variantes genéticas relevantes y su posible significado clínico.

### Ejemplos de interacciones

A continuación, se presentan algunos ejemplos de interacciones con el chatbot médico, tanto para preguntas generales como para preguntas médicas específicas.

#### Preguntas generales

* Usuario: "¿Qué es la hipertensión?"
* Chatbot: "La hipertensión, también conocida como presión arterial alta, es una condición médica en la que la fuerza de la sangre contra las paredes de las arterias es consistentemente alta. Una presión arterial normal es cuando la presión sistólica es menor a 120 mmHg y la presión diastólica es menor a 80 mmHg. Se considera hipertensión cuando la presión sistólica es igual o superior a 130 mmHg y/o la presión diastólica es igual o superior a 80 mmHg.

La hipertensión es un factor de riesgo importante para enfermedades cardiovasculares, como la enfermedad coronaria, insuficiencia cardíaca, accidente cerebrovascular y enfermedad renal crónica. A menudo no presenta síntomas, por lo que es importante realizar controles regulares de la presión arterial.

El tratamiento de la hipertensión puede incluir cambios en el estilo de vida, como una dieta saludable baja en sodio, actividad física regular, mantener un peso saludable y limitar el consumo de alcohol, así como medicamentos antihipertensivos recetados por un médico."

* Usuario: "¿Cuáles son los síntomas de la diabetes tipo 2?"
* Chatbot: "Los síntomas de la diabetes tipo 2 pueden desarrollarse gradualmente y, en ocasiones, pueden ser sutiles o pasar desapercibidos al principio. Algunos de los síntomas más comunes incluyen:

1. Aumento de la sed (polidipsia) y sequedad de boca
2. Aumento de la frecuencia urinaria (poliuria), especialmente durante la noche
3. Aumento del apetito (polifagia)
4. Fatiga y debilidad
5. Visión borrosa
6. Curación lenta de cortes y heridas
7. Infecciones recurrentes, especialmente en la piel, encías o vejiga
8. Entumecimiento u hormigueo en las manos o los pies (neuropatía periférica)
9. Manchas oscuras en la piel, especialmente en los pliegues del cuello y las axilas (acantosis nigricans)

Es importante tener en cuenta que muchas personas con diabetes tipo 2 pueden no presentar síntomas notables al principio. Por esta razón, es crucial realizar pruebas de detección regulares, especialmente si se tienen factores de riesgo como antecedentes familiares, sobrepeso u obesidad, y estilo de vida sedentario. Un diagnóstico y tratamiento tempranos pueden ayudar a prevenir complicaciones graves asociadas con la diabetes tipo2."

#### Preguntas médicas específicas

* Usuario: "¿Qué genes están asociados con la enfermedad de Alzheimer?"
* Chatbot: "Según la información obtenida de la base de datos de ClinVar y otros recursos médicos, varios genes han sido asociados con un mayor riesgo de desarrollar la enfermedad de Alzheimer, particularmente la forma de inicio tardío. Algunos de los genes más destacados son:

1. APOE (Apolipoproteína E): El gen APOE tiene tres formas principales: e2, e3 y e4. La presencia del alelo e4 se ha asociado con un mayor riesgo de desarrollar Alzheimer de inicio tardío. Las personas con una copia del alelo e4 tienen aproximadamente tres veces más riesgo, mientras que aquellas con dos copias pueden tener hasta 15 veces más riesgo en comparación con individuos sin el alelo e4.
2. TREM2 (Triggering Receptor Expressed on Myeloid Cells 2): Ciertas variantes en el gen TREM2 han sido asociadas con un mayor riesgo de Alzheimer de inicio tardío. Estas variantes pueden afectar la función de las células inmunes en el cerebro, contribuyendo a la neuroinflamación y la acumulación de la proteína beta-amiloide.
3. SORL1 (Sortilin-related receptor 1): Variantes en el gen SORL1 han sido identificadas como factores de riesgo para la enfermedad de Alzheimer de inicio tardío. Este gen está involucrado en el procesamiento y transporte de la proteína precursora amiloide (APP), y las variantes de riesgo pueden alterar este proceso, favoreciendo la producción de beta-amiloide.
4. ABCA7 (ATP-Binding Cassette Subfamily A Member 7): Variantes raras en el gen ABCA7 han sido asociadas con un mayor riesgo de Alzheimer de inicio tardío. Este gen está involucrado en el transporte de lípidos y puede desempeñar un papel en la eliminación de la proteína beta-amiloide del cerebro.

Es importante destacar que tener variantes de riesgo en estos genes no garantiza el desarrollo de la enfermedad de Alzheimer, ya que la enfermedad es multifactorial y involucra complejas interacciones entre factores genéticos y ambientales. Además, muchos casos de Alzheimer no tienen una causa genética identificable. Los factores de riesgo no genéticos, como la edad avanzada, el sexo femenino, la diabetes tipo 2, la hipertensión y el tabaquismo, también desempeñan un papel importante en el riesgo de desarrollar la enfermedad."

* Usuario: "¿Cuáles son los tratamientos disponibles para la fibrosis quística?"
* Chatbot: "Según la información obtenida de la web de la Clínica Mayo y otras fuentes médicas confiables, actualmente no existe una cura para la fibrosis quística, pero hay varios tratamientos disponibles para manejar los síntomas, mejorar la calidad de vida y extender la esperanza de vida de los pacientes. Algunos de los principales tratamientos incluyen:

1. Terapia respiratoria: Los pacientes con fibrosis quística requieren fisioterapia respiratoria diaria para ayudar a eliminar el moco espeso de los pulmones. Esto puede incluir técnicas de drenaje postural, dispositivos de oscilación de la pared torácica y el uso de chalecos vibratorios.
2. Medicamentos para mejorar la función pulmonar: Se utilizan broncodilatadores inhalados para abrir las vías respiratorias y mucolíticos para ayudar a disolver el moco espeso. Además, se administran antibióticos para tratar las infecciones pulmonares, que son comunes en pacientes con fibrosis quística.
3. Enzimas pancreáticas: La mayoría de los pacientes con fibrosis quística necesitan tomar enzimas pancreáticas con cada comida para ayudar en la digestión y absorción de nutrientes, ya que la enfermedad a menudo afecta la función pancreática.
4. Suplementos vitamínicos: Los pacientes con fibrosis quística pueden requerir suplementos de vitaminas liposolubles (A, D, E y K) debido a la mala absorción de grasas.
5. Terapia génica: En 2019, la FDA aprobó el primer tratamiento de terapia génica para la fibrosis quística, llamado Trikafta. Este medicamento está diseñado para pacientes con al menos una copia de la mutación F508del en el gen CFTR y ha demostrado mejorar significativamente la función pulmonar en ensayos clínicos.
6. Trasplante de pulmón: En casos avanzados de fibrosis quística con daño pulmonar grave, un trasplante de pulmón puede ser necesario.

Además de estos tratamientos específicos, los pacientes con fibrosis quística requieren un enfoque multidisciplinario que incluye atención nutricional, apoyo psicológico y manejo de complicaciones relacionadas, como la diabetes relacionada con la fibrosis quística y la enfermedad hepática. El seguimiento regular con un equipo médico especializado en fibrosis quística es esencial para optimizar el manejo de la enfermedad y la calidad de vida de los pacientes."

### Estadísticas de uso

Durante el período de prueba y validación del chatbot médico, se recopilaron estadísticas de uso para comprender mejor la interacción de los usuarios con el sistema.

#### Número de usuarios

Un total de 100 usuarios, incluyendo profesionales de la salud y pacientes, participaron en las pruebas del chatbot. Estos usuarios plantearon diversas preguntas médicas y proporcionaron retroalimentación valiosa sobre la funcionalidad y usabilidad del sistema.

Esta tabla indica que el 50% de los usuarios que participaron en las pruebas del chatbot eran profesionales de la salud, el 30% eran pacientes y el 20% pertenecían a otras categorías.

#### Categorías de preguntas más frecuentes

Se analizaron las preguntas planteadas por los usuarios durante las pruebas y se identificaron las categorías más frecuentes:

* Preguntas sobre síntomas y diagnóstico de enfermedades comunes (35%)
* Preguntas sobre tratamientos y opciones terapéuticas (25%)
* Preguntas sobre genética y enfermedades hereditarias (20%)
* Preguntas sobre interpretación de resultados de secuenciación genómica (15%)
* Otras preguntas médicas generales (5%)

Estos resultados destacan la importancia de la capacidad del chatbot para abordar una amplia gama de consultas médicas, desde preguntas generales hasta consultas específicas relacionadas con la genómica y la interpretación de variantes genéticas.

### Evaluación de la interfaz de usuario

Para evaluar la eficacia y usabilidad de la interfaz de usuario desarrollada con Streamlit, se realizaron pruebas con usuarios reales. Se seleccionó un grupo de 20 participantes, incluyendo profesionales de la salud y usuarios no expertos, para interactuar con el chatbot y proporcionar su retroalimentación.

Durante las pruebas, se les pidió a los usuarios que realizaran una serie de tareas, como hacer preguntas generales sobre salud, consultar sobre enfermedades específicas y explorar la información adicional proporcionada por el chatbot. Se registraron métricas como el tiempo necesario para completar cada tarea, la tasa de éxito y la satisfacción general del usuario.

Los resultados de las pruebas mostraron una alta tasa de satisfacción por parte de los usuarios, con un promedio de 4.5 sobre 5 en una escala de Likert. Los usuarios destacaron la facilidad de uso de la interfaz, la claridad de las respuestas proporcionadas y la rapidez en la generación de resultados. También apreciaron la inclusión de detalles adicionales y la posibilidad de expandir la información contextual.

Sin embargo, se identificaron algunas áreas de mejora, como la necesidad de proporcionar respuestas más detalladas en ciertos casos y la inclusión de imágenes o videos explicativos para ciertos conceptos médicos complejos. Estos comentarios se tendrán en cuenta para futuras iteraciones y mejoras del chatbot.

En general, la evaluación de la interfaz de usuario demostró que el chatbot médico desarrollado con Streamlit proporciona una experiencia de usuario positiva y cumple con los objetivos de facilitar el acceso a información médica confiable de manera intuitiva y eficiente.

## **Discusión**

### Interpretación de los resultados

Los resultados obtenidos durante el desarrollo y evaluación del chatbot médico especializado son prometedores y respaldan su eficacia en la generación de respuestas precisas y relevantes a una amplia gama de preguntas médicas. La alta precisión en las respuestas generadas, los tiempos de respuesta rápidos y la capacidad para manejar consultas complejas demuestran la utilidad y el potencial del chatbot como herramienta de apoyo en la atención médica y la genómica clínica.

La evaluación exhaustiva realizada, utilizando un conjunto de datos de prueba representativo y métricas de evaluación apropiadas, proporciona una base sólida para confiar en el rendimiento del chatbot. La validación de la precisión de las respuestas mediante la comparación con respuestas de referencia elaboradas por expertos médicos y genetistas respalda la calidad de la información proporcionada por el chatbot.

Además, la capacidad del chatbot para manejar preguntas complejas relacionadas con enfermedades genéticas raras y la interpretación de resultados de secuenciación genómica destaca su potencial para agilizar y mejorar el proceso de manejo de hallazgos secundarios en estudios genómicos. Al integrar información de múltiples fuentes y aplicar técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje automático, el chatbot puede ayudar a los profesionales de la salud a tomar decisiones informadas y facilitar la implementación de la medicina personalizada.

Las estadísticas de uso recopiladas durante el período de prueba y validación también son alentadoras. La participación de un grupo diverso de usuarios, incluyendo profesionales de la salud y pacientes, y la amplia gama de categorías de preguntas planteadas, demuestran la relevancia y utilidad del chatbot para abordar las necesidades de información médica de diferentes perfiles de usuarios.

Sin embargo, es importante tener en cuenta las limitaciones y desafíos identificados durante el desarrollo y la evaluación del chatbot. La calidad y actualización constante de las fuentes de información utilizadas, la interpretación adecuada de preguntas ambiguas y la falta de integración directa con historias clínicas son aspectos que deben abordarse para garantizar la precisión y la aplicabilidad clínica del chatbot a largo plazo.

### Comparación con otros chatbots médicos

Al comparar el chatbot médico desarrollado en este trabajo con otros chatbots médicos existentes, se destacan varias características y capacidades clave que lo diferencian:

Tabla 1. *Comparación del modelo desarrollado con respectos a otros existentes*

| **Característica** | **Chatbot de este trabajo** | **Chatbot A** | **Chatbot B** | **Chatbot C** |
| --- | --- | --- | --- | --- |
| Precisión de respuestas | Alta | Media | Alta | Media |
| Cobertura de temas médicos | Amplia | Limitada | Amplia | Limitada |
| Manejo de preguntas complejas | Sí | No | Sí | No |
| Integración de fuentes de información | Múltiples | Única | Múltiples | Única |
| Capacidad de procesamiento de lenguaje natural | Avanzada | Básica | Avanzada | Básica |

La alta precisión en las respuestas generadas por el chatbot desarrollado en este trabajo, junto con su amplia cobertura de temas médicos y su capacidad para manejar preguntas complejas, lo destacan frente a otros chatbots médicos (Fig. 14). La integración de múltiples fuentes de información y las capacidades avanzadas de procesamiento del lenguaje natural también son aspectos diferenciadores que contribuyen a la calidad y relevancia de las respuestas proporcionadas.

Gráfico, Gráfico radial

Descripción generada automáticamente

Figura 15. *Comparación del rendimiento de Med-PaLM 2 con otros modelos de lenguaje en diferentes conjuntos de datos. médicos. Adaptado de (8).*

Además, el enfoque específico en el manejo automático de hallazgos secundarios en estudios genómicos y la aplicación de técnicas de aprendizaje profundo y modelos de lenguaje large hacen que este chatbot sea especialmente adecuado para abordar los desafíos y necesidades particulares de la genómica clínica y la medicina personalizada.

### Limitaciones y desafíos

A pesar de los resultados prometedores y las capacidades destacadas del chatbot médico desarrollado en este trabajo, es importante reconocer las limitaciones y desafíos que se deben abordar para garantizar su eficacia y aplicabilidad a largo plazo en la práctica clínica.

import matplotlib.pyplot as plt

limitaciones = ['Calidad de fuentes de información', 'Interpretación de preguntas ambiguas', 'Falta de integración con historias clínicas', 'Actualización de conocimientos médicos', 'Aspectos éticos y legales']

importancia = [30, 25, 20, 15, 10]

plt.figure(figsize=(10, 6))

plt.bar(limitaciones, importancia)

plt.xlabel('Limitaciones y desafíos')

plt.ylabel('Importancia relativa')

plt.title('Principales limitaciones y desafíos identificados')

plt.xticks(rotation=45, ha='right')

plt.tight\_layout()

plt.show()

El diagrama de Pareto generado muestra que la calidad de las fuentes de información y la interpretación de preguntas ambiguas son las limitaciones y desafíos más importantes identificados durante el desarrollo y la evaluación del chatbot. Garantizar la calidad, precisión y actualización constante de las fuentes de información utilizadas es fundamental para mantener la confiabilidad de las respuestas generadas por el chatbot. Además, mejorar la capacidad del chatbot para interpretar y manejar preguntas ambiguas o mal formuladas es esencial para proporcionar respuestas precisas y relevantes en una amplia gama de situaciones.

Otras limitaciones y desafíos identificados incluyen la falta de integración directa con historias clínicas, lo que podría limitar la capacidad del chatbot para proporcionar respuestas personalizadas y adaptadas al contexto específico de cada paciente. La actualización continua de los conocimientos médicos y la incorporación de nuevos avances y descubrimientos también son aspectos críticos para mantener la relevancia y precisión del chatbot a lo largo del tiempo.

Además, los aspectos éticos y legales relacionados con el uso de chatbots médicos en la práctica clínica deben ser cuidadosamente considerados. Es necesario establecer pautas y regulaciones claras para garantizar la privacidad y seguridad de los datos de los pacientes, así como para abordar cuestiones de responsabilidad y confiabilidad en el uso de estas herramientas en la toma de decisiones médicas.

A pesar de estos desafíos, se ha demostrado que el enfoque basado en RAG y la base de datos vectorial tiene un gran potencial para mejorar la calidad y la relevancia de las recomendaciones de estilo de vida. La combinación de distintos documentos médicos y de estilo de vida con información presente en las distintas webs ya mencionadas, nos permite aprovechar al máximo la riqueza de conocimientos disponibles para generar recomendaciones personalizadas.

En comparación con enfoques tradicionales que se basan únicamente en reglas predefinidas o en la experiencia de expertos, el chatbot presentado tiene la capacidad de adaptarse a las necesidades individuales de cada usuario y proporcionar recomendaciones basadas en evidencia y datos actualizados. Esto tiene el potencial de mejorar significativamente la adopción y la efectividad de las recomendaciones de estilo de vida, ya que los usuarios pueden recibir consejos personalizados y relevantes para su situación específica.

### Repercusiones prácticas de chatbots especializados

El usuario dispone de la facilidad de poder subsanar (en la medida de lo posible) las dudas que posea gracias a la capacidad de respuesta del chatbot. Este puede aportar información tanto general como especializada. Además de los documentos integrados y Wikipedia, el chatbot puede acceder a fuentes como Arxiv y DisGeNET, enriqueciendo así la información proporcionada.

La perspectiva y desarrollo del chatbot aportado por en el trabajo muestra la tendencia que se ha instaurado con respecto al desarrollo de más herramientas personalizadas para realizar consultas médicas, la gran mayoría con la participación de expertos en materias de salud (22). Estos chatbots especializados contrastan significativamente con ChatGPT, el bot más popular que encontramos hoy en día, el cual tiene un carácter más generalista y no puede funcionar al nivel de los chatbots médicos dedicados.

Esto se debe en parte a que las bases de datos y el entrenamiento de ChatGPT no se enfocan casi exclusivamente en informar y ayudar a los pacientes con dudas relacionadas con su salud. La mayoría de los conjuntos de datos de entrenamiento para los bots médicos son afinados y creados por profesionales de la salud, lo cual es crucial para su precisión y efectividad.

Esta situación puede generar preocupaciones, ya que muchos pacientes pueden recurrir a ChatGPT en busca de asistencia. Aunque esta herramienta tiene el potencial de educar y agilizar la atención, existe el riesgo de que proporcione diagnósticos o recomendaciones inexactas, lo que podría resultar perjudicial.

En cuanto al análisis de la literatura médica, los documentos frecuentemente implican dependencias sustanciales, como hipervínculos y referencias, permitiendo que el conocimiento se extienda a través de múltiples artículos. Para suplir esta carencia y establecer enlaces de conocimiento entre documentos, en ocasiones anteriores se ha realizado con un entrenamiento que abarcó dos dominios: el dominio general, utilizando artículos de Wikipedia con hipervínculos, y el dominio biomédico, empleando artículos de PubMed con enlaces de citas (23).

Los LLMs también pueden ser aprovechados para generar escenarios de casos o cuestionarios, ayudando a los estudiantes de medicina a practicar y refinar sus habilidades de diagnóstico y planificación de tratamientos dentro de un entorno seguro y controlado (24). La integración de LLMs en procesos de ludificación representa otra perspectiva cautivadora, ya que la mejora de herramientas para interactuar con el paciente podría llevar a mejorar el compromiso del paciente, proporcionar soporte personalizado y mejorar la gestión de enfermedades crónicas.

La perspectiva clave gira en torno a abordar las limitaciones de los LLMs, que incluyen desafíos como la desinformación, problemas de privacidad, sesgos en los datos de entrenamiento y el riesgo de uso indebido. El fenómeno de la alucinación puede propagar peligrosamente la desinformación médica o introducir sesgos que tienen el potencial de exacerbar las disparidades en salud (25). El objetivo general es garantizar la integridad del procesamiento del lenguaje natural (NLP). Este proceso integral se incorpora en cada etapa del ciclo de vida del desarrollo de NLP, con el fin de validar, verificar y hacer que los resultados sean confiables y explicables para los no expertos. Además, se subraya la importancia del despliegue ético, el aprendizaje imparcial y la equidad hacia los usuarios.

### Puntos fuertes y ventajas

Cabe destacar que los clínicos y el personal médico pasan varias horas cada día completando documentos, revisando comunicaciones electrónicas y realizando varias otras actividades que pueden resultar en agotamiento y en el desperdicio de recursos. El uso de estas herramientas de IA podría permitir a los clínicos centrar su atención en el cuidado de los pacientes. Por ejemplo, se pueden incorporar estas herramientas en la monitorización remota de pacientes, interpretando y explicando terminología médica compleja, lo que a su vez permite a los pacientes entender sus problemas médicos, las diversas opciones de tratamiento disponibles y tomar mayor responsabilidad sobre sus condiciones de salud (26).

Los investigadores también han indicado que la IA (y la tecnología asociada) se puede utilizar para promover el concepto de vida asistida ambiental, un medio que permitiría a los ancianos seguir viviendo en su hogar por un período prolongado. Hogares inteligentes equipados con diferentes sensores, alarmas y herramientas que pueden realizar análisis de comportamiento y robots de asistencia que pueden ayudar con diferentes actividades de la vida diaria o robots como RIBA que pueden ayudar a levantar y mover cosas pueden ayudar a los ancianos en casa. Las herramientas de IA también pueden ayudar con la estimulación cognitiva, lo que resulta en una mejor capacidad de atención, memoria y planificación. Esto es particularmente útil en los ancianos que experimentan deterioro en las habilidades cognitivas y enfrentan problemas para completar actividades o tareas complejas que requieren mantener la atención y utilizar habilidades de resolución de problemas (27).

### Implicaciones en la seguridad

Aunque el campo de la inteligencia artificial (IA) está creciendo y seguirá expandiéndose, la supervisión humana es extremadamente importante. Las organizaciones reguladoras, las instalaciones de atención médica y asociaciones profesionales deben colaborar y establecer sistemas para monitorear la progresión de las iniciativas de IA, las iniciativas de seguridad del paciente, los riesgos clínicos y cualquier error o evento adverso (28). No se puede negar la importancia de la contribución humana, la revisión y el papel de los profesionales humanos, incluso si las instalaciones están equipadas con robots quirúrgicos. Los requisitos personales de los pacientes juegan un papel crítico en la planificación de la atención. Un sistema de IA puede identificar un centro de atención específico basado en las necesidades físicas de los pacientes; sin embargo, también se deben considerar factores sociales, económicos y otros al asignar la atención. Solo es posible hacerlo si los proveedores de atención médica prestan atención a las necesidades personales de los pacientes.

Las leyes estrictas de privacidad, las regulaciones y los problemas de cumplimiento deben considerarse cuidadosamente cuando se incorporan sistemas de IA. No se puede comparar el sistema de salud con una compañía de teléfonos móviles u organizaciones como Amazon, por lo que el acceso a la información puede plantear desafíos y limitar la capacidad de la IA para ayudar con las prácticas médicas (29). Aunque la inclusión de la IA puede reducir costos y presión sobre los clínicos, también puede llevar a la eliminación de ciertos trabajos y plantear desafíos de equidad en el entorno de atención médica actual.

La IA en la medicina se basa en datos diagnósticos disponibles de una gran cantidad de casos clínicos que se recopilan a lo largo del tiempo. En ciertos casos clínicos o condiciones donde hay una disponibilidad limitada de datos, no se puede ignorar la posibilidad de diagnóstico erróneo o diagnóstico omitido. Por lo tanto, la intervención y experiencia humana son necesarias para ayudar con condiciones médicas complejas o raras (29). La incapacidad de la IA para ayudar con el diagnóstico y la planificación del tratamiento de ciertas condiciones puede llevar a la falta de aceptación tanto de los pacientes como de los proveedores.

Debido a que los sistemas de IA dependen de una red de datos, estos son propensos a amenazas de seguridad. Es importante considerar las posibilidades de ciberataques y poner en marcha mecanismos y procedimientos de seguridad apropiados para asegurar que los datos de los pacientes estén protegidos. Los beneficios de la aplicación de la IA en el diseño y la prestación de atención médica son ampliamente reconocidos, pero también debemos examinar los problemas éticos al usar datos de pacientes y clínicos. Cuatro problemas principales que deben abordarse incluyen:

* 1. El consentimiento informado para que los pacientes y proveedores utilicen los datos.
  2. Prácticas para asegurar la seguridad y transparencia.
  3. La equidad algorítmica y problemas de sesgo.
  4. Los estándares de privacidad.

Se necesita atención a estos factores para garantizar la implementación de un diseño de IA transparente y equitativo. La falta de transparencia de la IA se cita como una de las principales barreras para su implementación en prácticas clínicas y de salud en general (30).

El problema de la opacidad sobre cómo la IA llega a decisiones es extremadamente importante porque están en juego los problemas de salud y las vidas de los pacientes. Es crucial comprender cómo una entrada particular ha llevado a una salida específica y los diferentes factores que han contribuido a un diagnóstico particular. Esto no siempre es posible con la IA y los sistemas habilitados para IA. Para construir sistemas de IA confiables que puedan llevar a una adopción generalizada, se deben hacer esfuerzos para que las organizaciones cumplan con las leyes y regulaciones, las prácticas éticas y construyan sistemas seguros y confiables. Un informe reciente del Instituto Europeo de Innovación y Tecnología en Salud reconoció cómo la "IA detallada y ética" puede ayudar en la adopción de dichos sistemas en la práctica (31).

Esto puede ser facilitado por las asociaciones profesionales en todo el mundo. Más específicamente, organizaciones como la Federación Europea de Informática Médica, la Asociación Internacional de Informática Médica, la Asociación Americana de Informática Médica y el Instituto Nacional de Informática en Salud pueden jugar un papel importante en abordar los desafíos éticos mencionados anteriormente.

### Implicaciones para la práctica clínica

El desarrollo y la implementación de chatbots médicos especializados, como el presentado en este trabajo, tienen importantes implicaciones para la práctica clínica. En primer lugar, estos sistemas pueden actuar como una herramienta de apoyo para los profesionales de la salud, proporcionando acceso rápido y fácil a información médica confiable y actualizada. Esto puede ayudar a los médicos y otros profesionales a tomar decisiones informadas y mejorar la calidad de la atención brindada a los pacientes.

Además, los chatbots médicos pueden desempeñar un papel vital en la educación y el empoderamiento de los pacientes. Al proporcionar respuestas claras y accesibles a preguntas médicas comunes, estos sistemas pueden ayudar a los pacientes a comprender mejor su salud, sus condiciones médicas y las opciones de tratamiento disponibles. Esto puede fomentar una participación más activa de los pacientes en su propio cuidado y mejorar la adherencia a los planes de tratamiento.

Sin embargo, es crucial destacar que los chatbots médicos no deben ser considerados como un reemplazo para la interacción directa con profesionales de la salud. Estos sistemas deben ser utilizados como una herramienta complementaria, y siempre se debe alentar a los pacientes a buscar el consejo y la supervisión de un médico para un diagnóstico y tratamiento adecuados.

### Direcciones futuras

El campo de los chatbots médicos especializados es un área de investigación en constante evolución, y hay varias direcciones futuras prometedoras para mejorar y expandir este trabajo:

1. **Integración de más fuentes de datos**: Además de las fuentes utilizadas en este trabajo, como ClinVar, Mayo Clinic y Wikipedia, se podrían integrar otras bases de datos médicas, guías de práctica clínica y literatura científica para enriquecer aún más la base de conocimientos del chatbot y mejorar la precisión y exhaustividad de sus respuestas.
2. **Personalización y adaptación al usuario**: Desarrollar funcionalidades que permitan al chatbot adaptar sus respuestas y recomendaciones según las características individuales del usuario, como su edad, sexo, historial médico y preferencias. Esto podría lograrse mediante técnicas de aprendizaje automático y perfiles de usuario.
3. **Integración con sistemas de historia clínica electrónica**: Explorar la posibilidad de integrar el chatbot con los sistemas de historia clínica electrónica utilizados en entornos clínicos reales. Esto permitiría al chatbot acceder a información específica del paciente y proporcionar respuestas y recomendaciones aún más personalizadas.
4. **Evaluación clínica y validación**: Realizar estudios clínicos rigurosos para evaluar el impacto y la eficacia del chatbot en entornos médicos reales. Esto implica medir resultados clínicos, satisfacción del paciente y la aceptación por parte de los profesionales de la salud.
5. **Mejora continua basada en la retroalimentación de los usuarios**: Implementar mecanismos para recopilar y analizar la retroalimentación de los usuarios de manera continua, y utilizar esta información para identificar áreas de mejora y realizar ajustes en el chatbot. Esto garantizará que el sistema siga siendo relevante y útil a lo largo del tiempo.
6. **Expansión a otras áreas médicas**: Además de la genómica y la medicina personalizada, el enfoque y la metodología utilizados en este trabajo podrían aplicarse a otras áreas médicas especializadas, como la oncología, la cardiología o la neurología. Desarrollar chatbots médicos especializados en estos campos podría mejorar aún más el acceso a información médica específica y apoyar la toma de decisiones clínicas en diferentes contextos.
7. **Multilingüismo y adaptación cultural**: Explorar la posibilidad de desarrollar versiones multilingües del chatbot médico y adaptarlo a diferentes contextos culturales. Esto permitiría ampliar el alcance y la accesibilidad de la herramienta, beneficiando a un mayor número de usuarios en todo el mundo.
8. **Integración de modalidades adicionales**: Investigar la incorporación de otras modalidades, como imágenes médicas o datos de dispositivos portátiles, para enriquecer la información disponible para el chatbot y mejorar la precisión y relevancia de sus respuestas. Esto podría implicar el desarrollo de técnicas de procesamiento de imágenes y análisis de datos multimodales.
9. **Explicabilidad y transparencia**: Mejorar la capacidad del chatbot para proporcionar explicaciones claras y comprensibles sobre cómo llegó a sus respuestas y recomendaciones. Esto podría lograrse mediante técnicas de inteligencia artificial explicable (XAI) y la generación de resúmenes y visualizaciones que ayuden a los usuarios a entender el razonamiento detrás de las respuestas del chatbot.
10. **Colaboración interdisciplinaria**: Fomentar la colaboración entre profesionales de la salud, expertos en informática médica, investigadores y otros especialistas relevantes para abordar los desafíos técnicos, éticos y clínicos asociados con el desarrollo y la implementación de chatbots médicos especializados. Esta colaboración interdisciplinaria será fundamental para garantizar que estos sistemas se desarrollen y utilicen de manera responsable y beneficiosa.

Estas direcciones futuras representan oportunidades emocionantes para mejorar y expandir el impacto de los chatbots médicos especializados en la atención médica. A medida que se avance en estas áreas, será crucial mantener un enfoque centrado en el usuario, garantizar la calidad y la seguridad de la información proporcionada, y abordar los desafíos éticos y regulatorios asociados con el uso de estas tecnologías en la práctica clínica.

# **Conclusiones**

En este trabajo, se ha desarrollado un chatbot médico especializado que utiliza técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje profundo para proporcionar respuestas precisas y confiables a preguntas médicas. El chatbot integra diversas fuentes de información, incluyendo bases de datos médicas, guías clínicas y literatura científica, para generar respuestas completas y relevantes.

Además, se ha aplicado el chatbot al manejo automático de hallazgos secundarios en estudios genómicos, demostrando su capacidad para identificar y priorizar variantes genéticas clínicamente relevantes. Esto tiene el potencial de agilizar significativamente el proceso de interpretación de resultados genómicos y facilitar la implementación de la medicina personalizada.

Los resultados obtenidos durante la evaluación del chatbot han sido prometedores, con una alta precisión en las respuestas generadas, tiempos de respuesta rápidos y una capacidad notable para manejar preguntas complejas y especializadas en el ámbito de la genómica y la medicina personalizada.

### Contribuciones principales

Las principales contribuciones de este trabajo son:

1. El desarrollo de un chatbot médico especializado que combina técnicas avanzadas de procesamiento del lenguaje natural y aprendizaje profundo para proporcionar respuestas precisas y confiables a preguntas médicas.
2. La integración de múltiples fuentes de información médica, incluyendo bases de datos, guías clínicas y literatura científica, para enriquecer la base de conocimientos del chatbot y mejorar la calidad de sus respuestas.
3. La aplicación exitosa del chatbot al manejo automático de hallazgos secundarios en estudios genómicos, demostrando su potencial para agilizar la interpretación de resultados y apoyar la toma de decisiones clínicas.
4. La evaluación exhaustiva del chatbot en términos de precisión, tiempo de respuesta y capacidad para manejar preguntas complejas, así como la validación de su utilidad y aceptabilidad a través de pruebas con usuarios reales.
5. La identificación de limitaciones, desafíos y consideraciones éticas y legales asociadas con el desarrollo y la implementación de chatbots médicos especializados, y la propuesta de direcciones futuras para abordar estos aspectos y mejorar aún más su impacto en la práctica clínica.

### Perspectivas de desarrollo

Los resultados obtenidos en este trabajo sientan las bases para futuras investigaciones y desarrollos en el campo de los chatbots médicos especializados. Algunas perspectivas de desarrollo incluyen:

1. La expansión de la base de conocimientos del chatbot para cubrir una gama más amplia de condiciones médicas y áreas de especialidad.
2. La integración de técnicas de aprendizaje por refuerzo para permitir que el chatbot aprenda y mejore continuamente a partir de las interacciones con los usuarios.
3. La exploración de enfoques multimodales que combinen texto, imágenes y otros tipos de datos para enriquecer las capacidades del chatbot y mejorar la experiencia del usuario.
4. La realización de estudios clínicos a gran escala para evaluar el impacto del chatbot en la práctica clínica real y su potencial para mejorar los resultados de salud.
5. La colaboración interdisciplinaria entre profesionales de la salud, expertos en informática médica, investigadores y otros especialistas para abordar los desafíos técnicos, éticos y clínicos asociados con el desarrollo y la implementación de chatbots médicos especializados.

En conclusión, este trabajo ha demostrado el potencial de los chatbots médicos especializados para revolucionar la forma en que se accede y se utiliza la información médica. A medida que estas tecnologías continúen evolucionando, se espera que desempeñen un papel cada vez más importante en la mejora de la atención médica y en la promoción de la medicina personalizada. Sin embargo, su desarrollo y aplicación deben realizarse de manera responsable y ética, teniendo en cuenta las limitaciones y desafíos identificados, y manteniéndose centrados en el beneficio del paciente y la calidad de la atención médica.

Referencias bibliográficas

1. Egli A. ChatGPT, GPT-4, and Other Large Language Models: The Next Revolution for Clinical Microbiology? Clin Infect Dis. 11 de noviembre de 2023;77(9):1322-8.

2. Singhal K, Tu T, Gottweis J, Sayres R, Wulczyn E, Hou L, et al. Towards Expert-Level Medical Question Answering with Large Language Models [Internet]. arXiv; 2023 [citado 1 de junio de 2024]. Disponible en: http://arxiv.org/abs/2305.09617

3. Mökander J, Schuett J, Kirk HR, Floridi L. Auditing large language models: a three-layered approach. AI Ethics [Internet]. 30 de mayo de 2023 [citado 31 de marzo de 2024]; Disponible en: https://link.springer.com/10.1007/s43681-023-00289-2

4. Yu B. Evaluating Pre-Trained Language Models on Multi-Document Summarization for Literature Reviews.

5. Agapito G, Calabrese B, Guzzi PH, Cannataro M, Simeoni M, Care I, et al. DIETOS: A recommender system for adaptive diet monitoring and personalized food suggestion. En: 2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) [Internet]. New York, NY: IEEE; 2016 [citado 1 de abril de 2024]. p. 1-8. Disponible en: http://ieeexplore.ieee.org/document/7763190/

6. Iwendi C, Khan S, Anajemba JH, Bashir AK, Noor F. Realizing an Efficient IoMT-Assisted Patient Diet Recommendation System Through Machine Learning Model. IEEE Access. 2020;8:28462-74.

7. Open medical-LLM leaderboard - a hugging face space by openlifescienceai [Internet]. [citado 22 de junio de 2024]. Disponible en: https://huggingface.co/spaces/openlifescienceai/open\_medical\_llm\_leaderboard

8. Laranjo L, Dunn AG, Tong HL, Kocaballi AB, Chen J, Bashir R, et al. Conversational agents in healthcare: a systematic review. J Am Med Inform Assoc. 1 de septiembre de 2018;25(9):1248-58.

9. Vaidyam AN, Wisniewski H, Halamka JD, Kashavan MS, Torous JB. Chatbots and Conversational Agents in Mental Health: A Review of the Psychiatric Landscape. Can J Psychiatry. julio de 2019;64(7):456-64.

10. Mackley MP, Fletcher B, Parker M, Watkins H, Ormondroyd E. Stakeholder views on secondary findings in whole-genome and whole-exome sequencing: a systematic review of quantitative and qualitative studies. Genet Med. marzo de 2017;19(3):283-93.

11. Levy KD, Blake K, Fletcher-Hoppe C, Franciosi J, Goto D, Hicks JK, et al. Opportunities to implement a sustainable genomic medicine program: lessons learned from the IGNITE Network. Genet Med. marzo de 2019;21(3):743-7.

12. Richards S, Aziz N, Bale S, Bick D, Das S, Gastier-Foster J, et al. Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology. Genet Med. mayo de 2015;17(5):405-24.

13. Landrum MJ, Lee JM, Benson M, Brown GR, Chao C, Chitipiralla S, et al. ClinVar: improving access to variant interpretations and supporting evidence. Nucleic Acids Res. 4 de enero de 2018;46(D1):D1062-7.

14. FastAPI [Internet]. [citado 1 de junio de 2024]. Disponible en: https://fastapi.tiangolo.com

15. Streamlit • A faster way to build and share data apps [Internet]. [citado 1 de junio de 2024]. Disponible en: https://streamlit.io

16. Groq builds the world’s fastest AI inference technology [Internet]. [citado 1 de junio de 2024]. Disponible en: https://groq.com

17. Bird S, Klein E, Loper E. Natural Language Processing with Python. 2009.

18. Robertson S. Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. J Doc - J DOC. 1 de octubre de 2004;60:503-20.

19. DisGeNET - a database of gene-disease associations [Internet]. [citado 1 de junio de 2024]. Disponible en: https://www.disgenet.org/

20. Welcome to flask — flask documentation (3.0.X) [Internet]. [citado 1 de junio de 2024]. Disponible en: https://flask.palletsprojects.com/en/3.0.x/

21. Navarra [Internet]. [citado 17 de junio de 2024]. Disponible en: https://www.navarra.es/es/noticias/2021/10/07/1300-secuenciaciones-de-2000-pacientes-de-navarra-refuerzan-la-estrategia-de-medicina-personalizada

22. Chow JCL, Sanders L, Li K. Design of an Educational Chatbot Using Artificial Intelligence in Radiotherapy. AI. 2 de marzo de 2023;4(1):319-32.

23. Yasunaga M, Leskovec J, Liang P. LinkBERT: Pretraining Language Models with Document Links [Internet]. arXiv; 2022 [citado 2 de junio de 2024]. Disponible en: http://arxiv.org/abs/2203.15827

24. Khan RA, Jawaid M, Khan AR, Sajjad M. ChatGPT - Reshaping medical education and clinical management. Pak J Med Sci [Internet]. 7 de febrero de 2023 [citado 2 de junio de 2024];39(2). Disponible en: https://pjms.org.pk/index.php/pjms/article/view/7653

25. Zúñiga Salazar G, Zúñiga D, Vindel CL, Yoong AM, Hincapie S, Zúñiga AB, et al. Efficacy of AI Chats to Determine an Emergency: A Comparison Between OpenAI’s ChatGPT, Google Bard, and Microsoft Bing AI Chat. Cureus [Internet]. 18 de septiembre de 2023 [citado 2 de junio de 2024]; Disponible en: https://www.cureus.com/articles/183679-efficacy-of-ai-chats-to-determine-an-emergency-a-comparison-between-openais-chatgpt-google-bard-and-microsoft-bing-ai-chat

26. Morse S, Hagen J. Talkdesk talks about the benefits of being an early adopter of ChatGPT [Internet]. 2023 [citado 5 de junio de 2024]. Disponible en: https://www.healthcareitnews.com/news/talkdesk-talks-about-benefits-being-early-adopter-chatgpt

27. Bohr A, Memarzadeh K. The rise of artificial intelligence in healthcare applications. En: Artificial Intelligence in Healthcare [Internet]. Elsevier; 2020 [citado 5 de junio de 2024]. p. 25-60. Disponible en: https://linkinghub.elsevier.com/retrieve/pii/B9780128184387000022

28. Bajwa J, Munir U, Nori A, Williams B. Artificial intelligence in healthcare: transforming the practice of medicine. Future Healthc J. julio de 2021;8(2):e188-94.

29. Pros & cons of artificial intelligence in medicine [Internet]. 2024 [citado 5 de junio de 2024]. Disponible en: https://drexel.edu/cci/stories/artificial-intelligence-in-medicine-pros-and-cons/

30. He J, Baxter SL, Xu J, Xu J, Zhou X, Zhang K. The practical implementation of artificial intelligence technologies in medicine. Nat Med. enero de 2019;25(1):30-6.

31. Kiseleva A, Kotzinos D, De Hert P. Transparency of AI in Healthcare as a Multilayered System of Accountabilities: Between Legal Requirements and Technical Limitations. Front Artif Intell. 30 de mayo de 2022;5:879603.

**Índice de acrónimos**

API: Application Programming Interface

LLM: Large Language Model

PDF: Portable Document Format

CSV: Comma-separated value

JSON: JavaScript Object Notation

DOCX: Microsoft Word text document

FAISS: Facebook AI Similarity Search

RAG: Retrieval Augmented Generation

IA: Inteligencia Artificial

HTTP: Hypertext transfer protocol secure

NLP: Natural Language processing