Asistente virtual interactivo para resolver consultas relacionadas con motocicletas mediante RAG

Interactive virtual assistant for addressing motorcycle-related queries using RAG

William Andrés Velasquez Ruiz[1], Ángel Andrés Martínez Oñate[2], Juan Pablo Hoyos Sánchez[3]

[1] Mechatronic Engineer, Universidad Nacional de Colombia sede De La Paz, La Paz Colombia.
https://orcid.org/0009-0009-5202-0031

[2] Mechatronic Engineer, Universidad Nacional de Colombia sede De La Paz, La Paz, Colombia.

https://orcid.org/0009-0003-7673-6065

[3] Doctor in Electronic Sciences, Universidad Nacional de Colombia sede De La Paz, La paz, Colombia.
https://orcid.org/0000-0002-1844-9127

jhoyoss@unal.edu.co

Cite this article as: W. Velasquez Ruiz, A. Martínez Oñate, J. Hoyos Sánchez "*Asistente virtual interactivo para resolver consultas relacionadas con motocicletas mediante RAG*", Prospectiva, Vol. 24 N° 1 2026

Recibido: 18/08/2025 / Aceptado: 20/11/2025

http://doi.org/ 10.15665/rp.v24i1.3828

RESUMEN

Este trabajo presenta el diseño e implementación de un asistente virtual interactivo orientado a la resolución de dudas técnicas sobre motocicletas, específicamente la Boxer CT100 KS. El sistema fue construido utilizando un enfoque de Retrieval-Augmented Generation (RAG) combinado con modelos de lenguaje de gran escala (LLMs), operando de manera completamente local a través de una interfaz web con un avatar 2D. La base de conocimientos se generó a partir de manuales técnicos, los cuales fueron procesados y almacenados en una base de datos vectorial. Se evaluaron múltiples combinaciones de modelos de embeddings y generativos mediante marcos como RAGAS y DeepEval, utilizando métricas como faithfulness, context precision y answer relevancy. Los resultados permitieron identificar configuraciones óptimas del sistema, donde las mejores destacaron en las métricas clave —como el modelo de embedding sentence-transformers y de lenguaje Llama-3.3-70b, que logró un *faithfulness* de 0.964 y *context precision* de 0.971 en RAGAS-Mistral, y el modelo de embedding intfloat/multilingual-e5-base y Llama-3.3-70b, que alcanzó un *answer relevancy* de 0.971 en DeepEval-Llama3—, demostrando la viabilidad de soluciones

personalizadas y privadas para asistencia técnica basada en IA. Se proponen mejoras mediante la incorporación de capacidades multimodales y la ampliación del corpus técnico.

Palabras clave: Asistente virtual, RAG, LLM, motocicletas, LangChain, evaluación automática, inteligencia artificial.

ABSTRACT

This work presents the design and implementation of an interactive virtual assistant designed to resolve technical inquiries about motorcycles, specifically the Boxer CT100 KS. The system was built using a Retrieval-Augmented Generation (RAG) approach, combined with large language models (LLMs), and operates entirely locally through a web interface with a 2D avatar. The knowledge base was generated from technical manuals, which were processed and stored in a vector database. Multiple combinations of embedding and generative models were evaluated using frameworks such as RAGAS and DeepEval, applying metrics like faithfulness, context precision, and answer relevancy. The results allowed the identification of optimal system configurations, where the best ones excelled in key metrics —such as the sentence-transformers embedding model combined with the Llama-3.3-70b language model, achieving a faithfulness score of 0.964 and context precision of 0.971 in RAGAS-Mistral, and the intfloat/multilingual-e5-base embedding model with Llama-3.3-70b, reaching an answer relevancy of 0.971 in DeepEval-Llama3—, demonstrating the feasibility of customized and private AI-based technical assistance solutions. Improvements are proposed through the incorporation of multimodal capabilities and the expansion of the technical corpus.

Keywords: Virtual assistant, RAG, LLM, motorcycles, LangChain, automatic evaluation, artificial intelligence.

1. Introducción

El desarrollo de asistentes virtuales ha experimentado un crecimiento significativo gracias a los avances en inteligencia artificial (IA), particularmente en modelos de lenguaje de gran escala (LLMs) y sistemas de Retrieval-Augmented Generation (RAG). Modelos como GPT-4 o Grok procesan preguntas complejas en lenguaje natural y entregan respuestas relevantes para el usuario. No obstante, su uso en dominios específicos enfrenta desafíos debido a la terminología técnica, lo cual ha sido abordado mediante sistemas RAG aplicados a la gestión del conocimiento industrial, alcanzando un MRR del 88% en servicios técnicos [1]. Asimismo, se ha demostrado que los LLMs pueden integrarse en sistemas de recomendación para capturar conocimiento de dominio abierto y mejorar la personalización en aplicaciones como la recomendación de productos [2].

El sistema RAG combina recuperación de información con generación de texto, utilizando una base de conocimientos (por ejemplo, catálogos de repuestos) para reducir alucinaciones del LLM y proporcionar respuestas contextualizadas. Su efectividad en entornos industriales ha sido demostrada mediante el uso de BM25 y embeddings para recuperar información técnica [1]. También se ha aplicado en asistentes de mantenimiento basados en ontologías OWL, combinando grafos de conocimiento con LMs para mejorar la precisión contextual en procedimientos técnicos, lo cual es relevante para la recomendación de repuestos

[3]. De forma similar, un enfoque híbrido KG-Vector RAG integró grafos de conocimiento con recuperación vectorial, logrando una precisión de coincidencia exacta del 77,8%, lo que refuerza su viabilidad en dominios técnicos [4]. Además, se ha empleado en control de calidad en manufactura, utilizando embeddings semánticos y reranking para diagnosticar defectos, con un enfoque transferible a la identificación de repuestos [5]. En contextos educativos, ha mejorado la precisión en respuestas a preguntas de libros de texto en un 9,84% [6]. Finalmente, la recuperación generativa (GCoQA) elimina la necesidad de índices vectoriales y mejora la recuperación en un 13,6%, ofreciendo una alternativa eficiente para este tipo de aplicaciones [7].

Se ha demostrado que la inteligencia artificial, incluyendo redes neuronales y aprendizaje por transferencia, mejora su precisión al mitigar la escasez de datos [8]. También se han utilizado LLMs para generar datos sintéticos que optimizan modelos de recuperación densa, un método aplicable a la creación de consultas sintéticas de repuestos en escenarios con información limitada [9]. El aprendizaje por refuerzo profundo (DRL) ha sido explorado para adaptar recomendaciones a preferencias dinámicas, lo cual permitiría personalizar sugerencias en entornos como talleres [10]. En el contexto del comercio electrónico, se identifican tendencias como la recuperación basada en contenido, especialmente relevantes para catálogos de repuestos [11]. Además, se ha observado que los usuarios expertos prefieren sistemas colaborativos, lo que sugiere combinar ambos enfoques para atender tanto a mecánicos como a usuarios inexpertos [12]. Finalmente, los LLMs pueden reforzar estos sistemas al integrar conocimiento externo, aportando una ventaja significativa para el asistente propuesto [2]

Las aplicaciones de asistentes virtuales y sistemas de recomendación en la literatura son diversas. En entornos industriales, se ha utilizado RAG para la gestión del conocimiento y el control de calidad, con enfoques aplicables a la recomendación de repuestos [1,5]. También se ha implementado un asistente virtual basado en RAG (RAGVA) en la gestión de carreteras, abordando desafíos de ingeniería como la escalabilidad y la evaluación, aspectos relevantes para el desarrollo del asistente virtual [13]. En el ámbito de la salud, RAG ha sido utilizado para extraer información clínica y responder preguntas neurológicas, lo que demuestra su versatilidad en dominios técnicos complejos [14,15]. Asimismo, se ha aplicado para generar resúmenes en lenguaje sencillo, una técnica útil para explicar repuestos a usuarios no especializados [16]. Finalmente, se han explorado factores que influyen en la adopción de asistentes de voz, resaltando su papel de apoyo, análogo al que se propone en talleres mecánicos [17].

Los asistentes virtuales enfrentan diversas barreras. Se han identificado preocupaciones relacionadas con la privacidad, la confianza y los costos en su adopción dentro del comercio minorista, aspectos también relevantes para usuarios en talleres mecánicos [18]. Estas inquietudes se complementan con los desafíos en la evaluación de sistemas RAG, donde se requiere el uso de métricas robustas para asegurar su fiabilidad en aplicaciones prácticas [13]. En este sentido, RAGAS (Retrieval Augmented Generation Assessment) surge como un framework innovador para la evaluación automatizada de sistemas RAG, sin necesidad de anotaciones humanas. RAGAS propone métricas estandarizadas como Context Precisión, Context Recall, Faithfulness y Answer correctness, que evalúan la relevancia y precisión de la recuperación de información, la fidelidad del modelo de lenguaje y la calidad de las respuestas generadas. Este enfoque es crucial en entornos como talleres mecánicos, donde la exactitud en la información sobre repuestos o procedimientos técnicos es vital para la confianza del usuario [19]. En contextos domésticos y vehiculares, se ha revisado el papel de los asistentes de voz proactivos, destacando que la capacidad de anticipar necesidades podría

mejorar la experiencia en entornos como talleres [20]. Finalmente, se ha subrayado la importancia de abordar aspectos éticos y de privacidad en sistemas de recomendación, lo que implica garantizar transparencia en el tratamiento de los datos de repuestos [21].

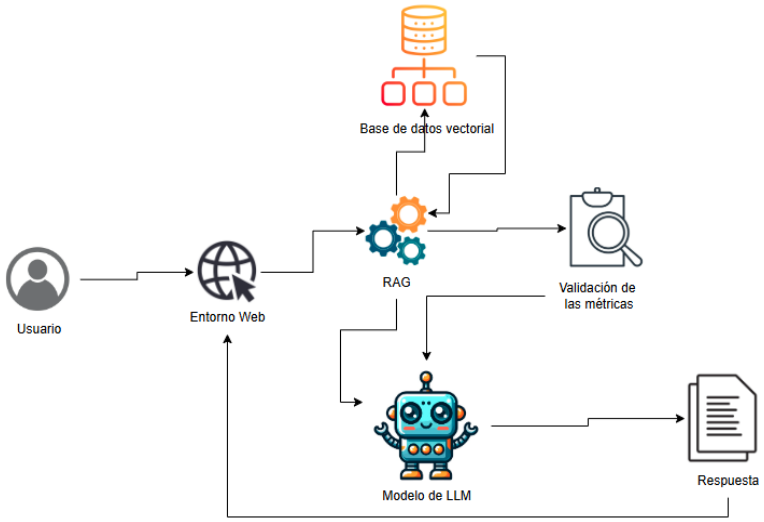
Este articulo propone un asistente virtual implementado mediante tecnologías web, utilizando un avatar 2D local que funciona directamente en el navegador. El objetivo fue recomendar repuestos y resolver dudas técnicas sobre motocicletas, particularmente la Boxer CT100 KS, mediante un sistema RAG respaldado por un modelo de lenguaje de gran escala (LLM). Para ello se desarrolló una base de conocimientos precisa y actualizada, lo cual implicó el procesamiento de manuales técnicos y documentos oficiales, que en escenarios futuros podría requerir colaboración directa con fabricantes de motocicletas. La compatibilidad de repuestos representó otro reto, similar a los problemas de precisión en sistemas de control de calidad, ya que ya que exigía una identificación exacta de componentes intercambiables y verificables a partir de catálogos técnicos limitados. También se consideraron aspectos críticos como la privacidad del usuario y la eficiencia del sistema, especialmente al ejecutarse de manera completamente local sin depender de servicios en la nube. Destacándose varias ventajas: (i) personalización del agente, habilitada por el uso de LLMs y recuperación aumentada por generación (RAG), (ii) arquitectura del sistema escalable hacia otras marcas o modelos de vehículos, con potencial de incorporar elementos visuales o tutoriales interactivos en futuras versiones, (iii) evaluación robusta del desempeño mediante RAGAS y DeepEval, y (iv) reproducibilidad y extensibilidad por parte de la comunidad científica.

En resumen, el flujo de trabajo propuesto, desde la extracción y segmentación de manuales técnicos con pdfplumber, la generación de embeddings multilingües y almacenamiento en Chroma, hasta la integración en LangChain con LLMs locales, interfaz en Flask y la evaluación mediante RAGAS y DeepEval, demostró una solución robusta, privada y de alto rendimiento para asistencia técnica en motocicletas, con configuraciones óptimas que superaron el 0.96 en métricas clave como fidelidad y precisión contextual.

2. Metodología

El desarrollo del presente proyecto se basa en la creación de un asistente virtual interactivo capaz de responder preguntas, orientado a la resolución de dudas y preguntas que tenga el usuario acerca de la moto Boxer ct100 KS, para lo cual se proprone el flujo de trabajo descrito en al Figura 1. Este sistema se apoyó en un enfoque de Retrieval-Augmented Generation (RAG), utilizando modelos de lenguaje de gran escala (LLMs) para proporcionar respuestas precisas, contextualizadas y con un lenguaje natural. La solución se implementó principalmente en Python, integrando un framework como LangChain que es un framework de código abierto diseñado para facilitar la construcción de aplicaciones que integran modelos de lenguaje con datos externos, permitiendo orquestar flujos complejos de interacción entre LLMs, bases de datos, APIs y documentos locales [22], junto a múltiples librerías y otros frameworks orientados al procesamiento de documentos, la generación de embeddings y la conversión de texto a voz.

Figura 1: Flujo de trabajo del asistente virtual interactivo**.**



El entorno visual y de interacción con el usuario fue desarrollado utilizando tecnologías web (HTML, CSS y JavaScript). Se diseñó un avatar animado en 2D que se ejecutan localmente en el navegador, sin depender de entornos gráficos externos. Este avatar respondía visualmente a los mensajes del asistente mediante animaciones que simulaban movimiento al hablar, las cuales se activaban durante la reproducción de audio. La voz del asistente se generó mediante la API de síntesis de voz nativa del navegador (SpeechSynthesis), lo cual permitió mantener toda la solución de manera local, sin requerir acceso a servicios externos de texto a voz. Finalmente, se llevó a cabo una comparación entre las distintas respuestas generadas por los modelos seleccionados, con el fin de clasificar la utilidad y viabilidad de las recomendaciones ofrecidas.

2.1. Extracción y procesamiento de información

El primer paso en la progresión fue el establecimiento de una base de conocimientos compuesta principalmente por manuales de usuario final junto a catálogos técnicos y guías de ventas de la motocicleta. Estos estaban en formato pdf y al ser tecnológicamente ricos tenían varias dificultades de extracción de información como consecuencia de su riqueza en imágenes, tablas y formatos no textuales.

Para enfrentar este reto, se hizo uso de la biblioteca pdfplumber, que facilita el desglose minucioso de texto y estructuras como tablas a partir de documentos en formato PDF [23]. Adicionalmente, se migraron todos los documentos que se encontraban almacenados en una ruta específica del proyecto y se hizo un proceso de segmentación (chunking) que involucró segmentar cada escritura en tramos de texto como máximo de 1000 caracteres, dejando un solapamiento de 100 caracteres entre tramos, lo cual funcionó como ventana de contexto entre un chunk y otro.

Además del texto plano, se utilizó un proceso auxiliar para desplegar las tablas incluidas en los documentos. Cada tabla se descompuso en filas, y sus valores fueron concatenados y normalizados sin perder su estructura informativa y evitando así pérdida de contenido relevante. El texto de cada página y sus tablas incluidas se unieron en un único conjunto de entrada que luego fue procesada como un objeto enriquecido con metadatos como es el caso del nombre del archivo, fuente y páginas en total. Esta estructura aseguró un

manejo más sólido de información que permitió trazabilidad fácil de las respuestas generadas y sin perder mucha información que resultó útil al momento de generar las respuestas.

2.2. Generación de la base de datos vectorial

Una vez que estaba extraída y segmentada toda la información, el siguiente paso involucró almacenar en una base de datos vectorial el conocimiento resultante. Para este fin se utilizó Chroma, que es un almacenador especializado en datos vectoriales. Chroma es capaz de sustituir texto en forma de vectores numéricos usando embeddings y así permite calcular la semántica de semejanza entre preguntas de usuario y pasajes de base de datos[24]. Básicamente Chroma es un sistema de base de datos vector que es capaz de procesar eficientemente búsqueda semántica sobre grandes cantidades de texto tanto estructurado como no estructurado y es particularmente valioso en sistemas RAG centrados en recuperación y generación de información técnica. Los embeddings fueron generados utilizando modelos integrados en LangChain, que permiten convertir cada fragmento textual en una representación matemática de alta dimensión, los utilizados en el proyecto son intfloat/multilingual-e5-base y sentence-transformers/paraphrase-multiling ual-MiniLM-L12-v2 utilizando técnicas como en este caso la de la similitud del coseno, la cual se representa de la siguiente forma:

$$Sim(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Donde A y B son vectores de alta dimensión generados a partir de texto natural mediante modelos de embedding. Esta es una métrica comúnmente empleada para medir la cercanía semántica entre textos representados como vectores. Estas representaciones fueron almacenadas en un repositorio persistente, donde pueden ser consultadas de manera rápida cada vez que el usuario realiza una pregunta sin necesidad de volver a crear todo el embedding. Gracias a este enfoque, el sistema es capaz de recuperar el fragmento más relevante desde la base de datos vectorial, sirviendo como insumo directo para la generación de respuestas por parte del modelo de lenguaje elegido.

2.3. Definir la configuración del modelo generativo y la definición del asistente

Una vez procesada la de base de datos de vectorial, utilizamos un generador de modelos de lenguaje (LLM), que pueda entender consultas en un idioma natural y dar respuestas claras y acoplables. A esa finalidad recurrimos modelos como gemini-2.0-flash, Llama 3.3-70b y gemma2-9b-it experimentalmente verificados. Se definió un prompt específico para configurar la personalidad del asistente virtual. Este prompt se encuentra en el GitHub del proyecto [25] y describe el rol del agente como un asesor técnico especializado en motocicletas especialmente en la Boxer CT100 KS, capacitado para responder preguntas sobre garantías, procedimientos de mantenimiento y dudas de localización de partes. Esta personalización permite generar respuestas más alineadas con el tono y la intención del proyecto, aportando valor al usuario final mediante un lenguaje técnico, claro y empático. En esta parte también agregamos un historial de las respuestas preguntas y respuestas formuladas anteriormente para que así el agente tenga contexto de los temas de los cuales se está hablando y evite ser redundante en cosas que ya han quedado claras, como también limitamos la cantidad de chunks que más se parezcan a la consulta del usuario para el anexo a la query la cual se le manda al modelo de lenguaje.

2.4. Entorno Virtual

El sistema fue implementado con una interfaz gráfica web construida sobre el framework Flask, permitiendo al usuario interactuar con el asistente de forma completamente local. A través de esta interfaz, el usuario puede formular sus preguntas por medio de texto escrito, enfocadas a dudas técnicas sobre la motocicleta Boxer CT 100 KS. Cada vez que se realiza una consulta, ésta es enviada al backend desarrollado en Python, donde se procesa mediante el RAG. La respuesta generada se muestra en pantalla dentro de la misma interfaz web, garantizando una experiencia fluida y directa. Además, el sistema incluye una funcionalidad de retroalimentación auditiva. Una vez generada la respuesta textual, esta se convierte en un archivo de audio en formato MP3, el cual se reproduce automáticamente en la interfaz mediante un reproductor HTML5. Esta característica otorga un componente más natural e inmersivo a la interacción con el asistente, asemejándose al comportamiento de un asesor técnico con voz propia.

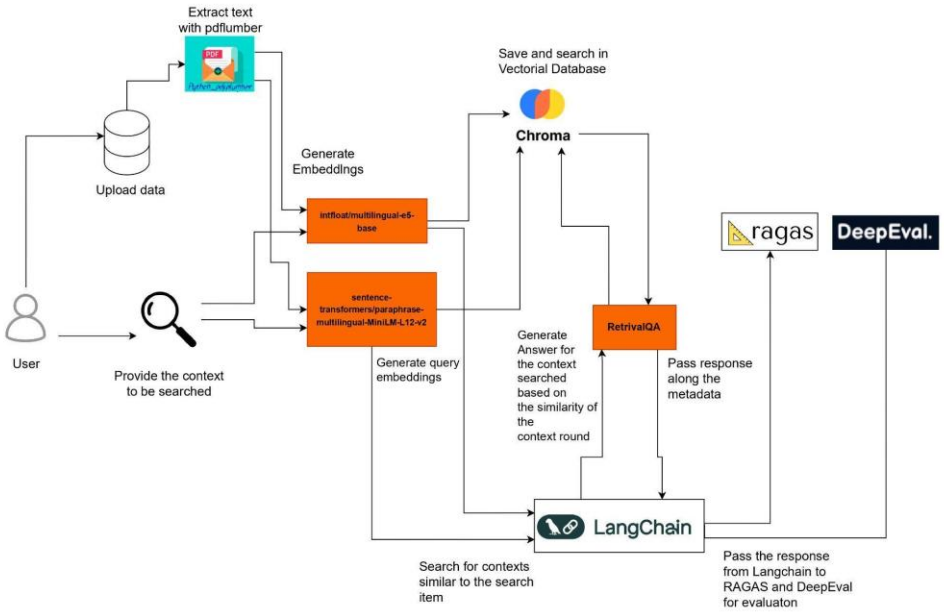
2.5. Evaluación y comparación de modelos

Por último, para autentificar el rendimiento del sistema, se propone realizar una evaluación comparativa entre los tres modelos lingüísticos, gemini-2.0-flash, Llama 3.3-70b y gemma2-9b-it. Para llevar a cabo esta evaluación, se utilizará el marco de evaluación RAGAS y DeepEval para analizar objetivamente el rendimiento de los sistemas basados en RAG [26], como se aprecia en la Figura 2. RAGAS permite analizar diferentes aspectos del sistema evaluando tanto la fase de recuperación de contexto como la de generación de respuestas[26]. Por su parte, DeepEval proporciona un marco robusto para evaluar modelos de lenguaje, enfocándose en métricas que complementan el análisis de RAGAS, permitiendo una evaluación integral del desempeño del sistema [26]. En este proyecto se utilizarán tres métricas clave proporcionadas por los marcos evaluadores: Faithfulness, Context precision y Answer relevancy.

- Faithfulness evalúa la fidelidad factual de las respuestas generadas, verificando que estén fundamentadas únicamente en el contexto recuperado y no contengan alucinaciones o información externa no sustentada.
- Context precision mide qué tan relevante y específico es el fragmento de información recuperado en relación con la pregunta formulada.
- Answer relevancy se centra en evaluar qué tan pertinente es la respuesta generada para el mensaje dado. Se asigna una puntuación más baja a las respuestas que están incompletas o contienen información redundante y las puntuaciones más altas indican una mejor relevancia.

Para la evaluación se construyó un conjunto de datos compuesto por 28 preguntas formuladas, centradas en aspectos técnicos y operativos de la motocicleta Boxer CT 100 KS. Cada entrada en el dataset incluye la pregunta del usuario, el contexto recuperado desde la base vectorial, la respuesta generada por el sistema y una respuesta esperada validada manualmente, la cual sirvió como punto de referencia para calcular las métricas deseadas, estas respuestas esperadas fueron formuladas de múltiples lecturas a la base.

Figura 2: Pipeline para la evaluación de sistema RAG



de conocimientos cargada al sistema como también el ingreso a un LLM más grande (GPT 4) luego se sintetizaron las respuestas para así poder tener las ground truths incorporadas en el conjunto de datos. Este conjunto será aplicado a las seis combinaciones posibles entre los modelos de embeddings y los modelos generativos (2 embeddings × 3 modelos generativos) en los dos marcos evaluativos, permitiendo una evaluación comparativa sistemática de cada configuración. El análisis resultante permitió identificar qué combinación de modelos ofrece un mejor equilibrio entre fidelidad, precisión contextual y exactitud de respuestas, proporcionando así evidencia empírica para orientar futuras decisiones de mejora en el diseño del sistema de asistencia virtual.

3. Resultados y discusión

Los resultados obtenidos corresponden a la evaluación de seis combinaciones entre dos modelos de embedding —E1: intfloat/multilingual-e5-base y E2: sentence-transformers/paraphrase-multilingual -MiniLM-L12-v2— con tres modelos de lenguaje —L1: Gemini-2.0-flash, L2: Llama-3.3-70b-versatile, y L3: gemma2-9b-it. El codigo desarrollado se encuentra disponible en el siguiente enlace al repositorio de GitHub: https://github.com/ElWilly9/AsistentedeMotosRAG.

La evaluación se realizó utilizando dos marcos: RAGAS y DeepEval, cada uno aplicado con dos modelos evaluadores distintos: Llama3 y Mistral. Las Tablas 1 y 2 muestran los puntajes obtenidos con RAGAS. En la evaluación con Llama3, la combinación E2L2 presenta el valor más alto en faithfulness (0.601), mientras que E1L2 alcanza el mejor resultado en answer relevancy (0.174). En cuanto a context precision, el mayor valor se obtiene con E1L3 (0.510). Por otro lado, bajo RAGAS con Mistral, E2L2 destaca en faithfulness

con un valor de 0.813, seguido de cerca por E1L2 con 0.784. Las combinaciones E2L1 (0.835) y E2L2 (0.822) logran las mayores puntuaciones en context precision. En answer relevancy, los valores son bajos en general, siendo E1L3(0.139) el más alto.

Las Tablas 3 y 4 presentan los resultados utilizando DeepEval. En este marco, se observa un desempeño superior en answer relevancy, donde E1L2 obtiene los mejores resultados con ambos evaluadores (0.810 con Llama3 y 0.795 con Mistral). En faithfulness, la combinación más destacada con Llama3 es E1L1 (0.530), mientras que con Mistral es E2L1 (0.465). La métrica context precision alcanza su valor máximo con E1L2 (0.415) usando Mistral.

Tabla 1: Evaluación de combinaciones de modelos de embedding(E) y LLM (L) para RAG usando RAGAS con Llama3:8b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	0.526	0.131	0.508
E1L2	0.525	**0.174**	0.319
E1L3	0.524	0.144	**0.510**
E2L1	0.511	0.122	0.354
E2L2	**0.601**	0.136	0.310
E2L3	0.563	0.104	0.354

Tabla 2: Evaluación de combinaciones de modelos de embedding(E) y LLM (L) para RAG usando RAGAS con Mistral:7b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	0.436	0.120	0.724
E1L2	0.784	0.137	0.801
E1L3	0.488	**0.139**	0.803
E2L1	0.468	0.106	**0.835**
E2L2	**0.813**	0.119	0.822
E2L3	0.393	0.096	0.762

Tabla 3: Evaluación de combinaciones de modelos de embedding(E) y LLM (L) para RAG usando DeepEval con Llama3:8b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	**0.530**	0.613	**0.421**

E1L2	0.414	**0.810**	0.271
E1L3	0.406	0.755	0.318
E2L1	0.387	0.709	0.338
E2L2	0.453	0.696	0.364
E2L3	0.452	0.565	0.276

Tabla 4: Evaluación de combinaciones de modelos de embedding(E) y LLM (L) para RAG usando DeepEval con Mistral:7b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	0.438	0.672	0.264
E1L2	0.251	**0.795**	**0.415**
E1L3	0.351	0.580	0.350
E2L1	**0.465**	0.728	0.301
E2L2	0.265	0.697	0.343
E2L3	0.286	0.643	0.288

Durante la evaluación, se detectaron errores en el cálculo de las métricas por parte de RAGAS y DeepEval, especialmente en preguntas con respuestas extensas o estructuras complejas, generando valores nulos o igualados a cero. Las tablas anteriores incluyen estos casos, por lo que reflejan el comportamiento general del sistema considerando también su fragilidad ante ciertas entradas.

3.1. Promedios sin errores de evaluación

Para mitigar los efectos de los errores mencionados, se calcularon promedios filtrados considerando únicamente las métricas válidas. Las Tablas 5 y 6 muestran los resultados bajo RAGAS. En ambas evaluaciones, todas las combinaciones alcanzan valores superiores a 0.76 en context precision, y en general se observan mejoras notables respecto a las tablas originales.

Tabla 5: Promedios filtrados para RAG usando RAGAS con Llama3:8b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	0.803	0.281	**0.908**
E1L2	0.767	0.269	0.843

E1L3	**0.882**	0.280	0.821
E2L1	0.707	0.203	0.766
E2L2	0.819	**0.293**	0.808
E2L3	0.861	0.214	0.810

Tabla 6: Promedios filtrados para RAG usando RAGAS con Mistral:7b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	0.823	0.255	0.950
E1L2	0.881	0.265	0.919
E1L3	0.944	0.281	0.961
E2L1	0.684	0.216	0.964
E2L2	**0.964**	**0.293**	**0.971**
E2L3	0.650	0.218	0.827

Con RAGAS y Llama3, E1L3 logra el mayor faithfulness (0.882), mientras que con Mistral, E2L2 destaca con 0.964. En ambos casos, los valores de context precision superan 0.80 en la mayoría de combinaciones.

En el caso de DeepEval (Tablas 7 y 8), los resultados también muestran una mejora significativa. Bajo Llama3, E1L1 obtiene el mayor faithfulness (0.922), y E1L2 alcanza 0.971 en answer relevancy. Bajo Mistral, varias combinaciones alcanzan valores perfectos de 1.000 en faithfulness, como E1L1, E1L3 y E2L3. En cuanto a context precision, las puntuaciones más altas se registran en E1L3 (0.905 con Mistral) y E1L1 (0.715 con Llama3). Estos resultados refuerzan la hipótesis de que el sistema presenta un buen desempeño cuando los evaluadores procesan correctamente las salidas.

En cuanto a context precision, las puntuaciones más altas se registran en E1L3 (0.905 con Mistral) y E1L1 (0.715 con Llama3). Estos resultados refuerzan la hipótesis de que el sistema presenta un buen desempeño cuando los evaluadores procesan correctamente las salidas.

Tabla 7: Promedios filtrados para RAG usando DeepEval con Llama3:8b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	**0.922**	0.845	**0.715**
E1L2	0.753	**0.971**	0.564
E1L3	0.717	0.866	0.593

E2L1	0.725	0.807	0.529
E2L2	0.765	0.671	0.490
E2L3	0.917	0.500	0.603

Tabla 8: Promedios filtrados para RAG usando DeepEval con Mistral:7b

Modelo	Faithfulness	Answer Relevancy	Context Presicion
E1L1	**1.000**	0.875	0.500
E1L2	0.754	0.819	0.637
E1L3	**1.000**	0.750	**0.905**
E2L1	0.833	**0.932**	0.715
E2L2	0.833	0.750	0.720
E2L3	**1.000**	0.792	0.543

En conjunto, los promedios filtrados ofrecen una estimación más realista del rendimiento del sistema, al eliminar el sesgo provocado por los errores en la evaluación. Además, brindan evidencia clara para tomar decisiones informadas sobre qué combinaciones de modelos utilizar en un entorno de producción.

En cuanto al desempeño, se observó que no existía una única combinación dominante de modelos. Sin embargo, algunas configuraciones, como E1L2 y E2L2, lograron destacar en diferentes métricas evaluadas por RAGAS y DeepEval, lo cual coincidió con hallazgos en la literatura sobre la importancia de ajustar tanto el modelo generativo como el modelo de recuperación para lograr un mejor balance entre fidelidad y precisión contextual [19,10]. Además, la comparación de métricas filtradas, excluyendo respuestas mal evaluadas, permitió una evaluación más realista del sistema y destacó su robustez ante consultas bien estructuradas.

No obstante, también se identificaron limitaciones importantes. Algunos errores sistemáticos de evaluación en las métricas automáticas dificultaron la comparación global entre combinaciones de modelos, particularmente en preguntas complejas o respuestas largas. Esta situación resalta la necesidad de contar con evaluaciones complementarias, incluyendo validación manual o el diseño de datasets de prueba más controlados. Asimismo, la calidad de las respuestas estuvo limitada por la calidad y cobertura de los documentos base, lo cual sugiere que la expansión de la base de conocimiento podría mejorar significativamente el desempeño del sistema.

Finalmente, el sistema se limitó a responder preguntas textuales con apoyo de voz, sin incorporar imágenes, diagramas ni funcionalidades interactivas más avanzadas. La incorporación de capacidades multimodales y mecanismos de retroalimentación de usuario serían pasos naturales en futuras iteraciones del sistema, tal como ha sido propuesto en trabajos recientes sobre asistentes virtuales proactivos y adaptativos [20,10].

4. Conclusiones

El desarrollo del asistente virtual propuesto demostró la viabilidad de integrar técnicas de Retrieval-Augmented Generation (RAG) con modelos de lenguaje de gran escala (LLMs) para la recomendación técnica de repuestos y la resolución de dudas en el dominio específico de motocicletas, en este caso la Boxer CT 100 KS. A través de la construcción de una base de conocimientos a partir de documentos técnicos y su transformación en una base de datos vectorial, fue posible recuperar información relevante y generar respuestas contextualizadas en lenguaje natural.

El sistema fue diseñado para operar de manera local, utilizando tecnologías web y un avatar animado en 2D que permite una experiencia de interacción fluida y accesible, sin requerir conexión a servicios externos. Esta decisión técnica también contribuyó a garantizar la privacidad del usuario y facilitar la ejecución del sistema en entornos con recursos limitados. Y, dada su naturaleza de código abierto, que facilita su extensibilidad hacia nuevas funcionalidades, se convierte en una alternativa a los sistemas propietarios.

Los experimentos realizados, utilizando múltiples combinaciones de modelos de embeddings y generativos, permitieron identificar dos configuraciones con mejor rendimiento en términos de fidelidad, relevancia contextual y precisión de respuesta. Además, el uso de marcos de evaluación como RAGAS y DeepEval ofreció una base objetiva y reproducible para valorar la calidad de las respuestas del sistema. La configuracion con mejore desempeño fue E2L2 (sentence-transformers/paraphrase-multilingual-MiniLM -L12-v2 + Llama-3.3-70b) con un faithfulness de 0.964 y context precision de 0.971 en RAGAS-Mistral, y la segunda fue E1L2 (intfloat/multilingual-e5-base + Llama-3.3-70b) con un answer relevancy de 0.971 en DeepEval-Llama3. Aunque existen limitaciones relacionadas con errores de evaluación en respuestas complejas, el sistema mostró un comportamiento robusto y un claro potencial para ser escalado a otros vehículos o marcas.

El sistema propuesto presentó varias limitaciones inherentes al enfoque adoptado. En primer lugar, los modelos de lenguaje empleados fueron predominantemente de código abierto, como Gemma2-9b-it y Llama-3.3-70b, con un número relativamente limitado de parámetros en comparación con alternativas propietarias de mayor escala, como GPT-4 o 5. Esta restricción en la capacidad de parámetros y en la complejidad estructural de los modelos pudo influir en la profundidad de las respuestas generadas, potencialmente incrementando la incidencia de alucinaciones o imprecisiones en consultas técnicas complejas, aunque se mitigó parcialmente mediante el uso de RAG. En segundo lugar, la base de conocimientos se vio confinada a datos públicos proporcionados por el fabricante, incluyendo manuales de usuario, carteles publicitarios y materiales de ventas, lo que excluyó información propietaria o detallada sobre componentes internos, diagnósticos avanzados o actualizaciones no divulgadas; esta limitación restringió la cobertura integral del sistema y su capacidad para abordar escenarios reales de mantenimiento o reparación. Finalmente, los marcos evaluadores utilizados, RAGAS y DeepEval, respaldados por modelos como Llama3 y Mistral, están sujetos a sesgos inherentes en sus algoritmos de puntuación, ya que dependen de interpretaciones subjetivas de lo que constituye una respuesta óptima o verídica, lo cual se evidenció en errores de cálculo observados durante la evaluación de respuestas extensas o complejas.

Como trabajo futuro, se propone ampliar la base de conocimientos con fuentes actualizadas directamente proporcionadas por fabricantes, incorporar capacidades multimodales para soportar imágenes o diagramas de piezas, y evaluar el sistema en escenarios reales con usuarios finales en talleres o entornos de servicio

técnico. Asimismo, podría integrarse un módulo de retroalimentación continua para mejorar el sistema de forma iterativa con base en las interacciones registradas.

Referencias

[1] L. C. Chen, M. S. Pardeshi, Y. X. Liao, and K. C. Pai, "Application of retrieval-augmented generation for interactive industrial knowledge management via a large language model," *Computer Standards & Interfaces*, vol. 94, p. 103995, Aug. 2025, doi: 10.1016/J.CSI.2025.103995.

[2] LinJianghao *et al.*, "How Can Recommender Systems Benefit from Large Language Models: A Survey," *ACM Transactions on Information Systems*, Jan. 2025, doi: 10.1145/3678004.

[3] H. Ludwig, T. Schmidt, and M. Kühn, "An ontology-based retrieval augmented generation procedure for a voice-controlled maintenance assistant," *Computers in Industry*, vol. 169, p. 104289, Aug. 2025, doi: 10.1016/J.COMPIND.2025.104289.

[4] Y. Wan, Z. Chen, Y. Liu, C. Chen, and M. Packianather, "Empowering LLMs by hybrid retrieval-augmented generation for domain-centric Q&A in smart manufacturing," *Advanced Engineering Informatics*, vol. 65, p. 103212, May 2025, doi: 10.1016/J.AEI.2025.103212.

[5] J. A. Heredia Álvaro and J. G. Barreda, "An advanced retrieval-augmented generation system for manufacturing quality control," *Advanced Engineering Informatics*, vol. 64, p. 103007, Mar. 2025, doi: 10.1016/J.AEI.2024.103007.

[6] H. A. Alawwad, A. Alhothali, U. Naseem, A. Alkhathlan, and A. Jamal, "Enhancing textual textbook question answering with large language models and retrieval augmented generation," *Pattern Recognition*, vol. 162, p. 111332, Jun. 2025, doi: 10.1016/J.PATCOG.2024.111332.

[7] Y. Li, N. Yang, L. Wang, F. Wei, and W. Li, "Generative retrieval for conversational question answering," *Information Processing & Management*, vol. 60, no. 5, p. 103475, Sep. 2023, doi: 10.1016/J.IPM.2023.103475.

[8] Q. Zhang, J. Lu, and Y. Jin, "Artificial intelligence in recommender systems," *Complex and Intelligent Systems*, vol. 7, no. 1, pp. 439–457, Feb. 2021, doi: 10.1007/S40747-020-00212-W/FIGURES/1.

[9] L. Silva and L. Barbosa, "Improving dense retrieval models with LLM augmented data for dataset search," *Knowledge-Based Systems*, vol. 294, p. 111740, Jun. 2024, doi: 10.1016/J.KNOSYS.2024.111740.

[10] X. Chen, L. Yao, J. McAuley, G. Zhou, and X. Wang, "Deep reinforcement learning in recommender systems: A survey and new perspectives," *Knowledge-Based Systems*, vol. 264, p. 110335, Mar. 2023, doi: 10.1016/J.KNOSYS.2023.110335.

[11] A. Valencia-Arias, H. Uribe-Bedoya, J. D. González-Ruiz, G. S. Santos, and E. C. Ramírez, "Artificial intelligence and recommender systems in e-commerce. Trends and research agenda,"

Intelligent Systems with Applications, vol. 24, p. 200435, Dec. 2024, doi: 10.1016/J.ISWA.2024.200435.

[12] S. Chinchanachokchai, P. Thontirawong, and P. Chinchanachokchai, "A tale of two recommender systems: The moderating role of consumer expertise on artificial intelligence based product recommendations," *Journal of Retailing and Consumer Services*, vol. 61, p. 102528, Jul. 2021, doi: 10.1016/J.JRETCONSER.2021.102528.

[13] R. Yang, M. Fu, C. Tantithamthavorn, C. Arora, L. Vandenhurk, and J. Chua, "RAGVA: Engineering retrieval augmented generation-based virtual assistants in practice," *Journal of Systems and Software*, vol. 226, p. 112436, Aug. 2025, doi: 10.1016/J.JSS.2025.112436.

[14] M. Alkhalaf, P. Yu, M. Yin, and C. Deng, "Applying generative AI with retrieval augmented generation to summarize and extract key clinical information from electronic health records," *Journal of Biomedical Informatics*, vol. 156, p. 104662, Aug. 2024, doi: 10.1016/J.JBI.2024.104662.

[15] L. Masanneck, S. G. Meuth, and M. Pawlitzki, "Evaluating base and retrieval augmented LLMs with document or online support for evidence-based neurology," *npj Digital Medicine*, vol. 8, no. 1, pp. 1–5, Dec. 2025, doi: 10.1038/S41746-025-01536-Y

[16] Y. Guo, W. Qiu, G. Leroy, S. Wang, and T. Cohen, "Retrieval augmentation of large language models for lay language generation," *Journal of Biomedical Informatics*, vol. 149, p. 104580, Jan. 2024, doi: 10.1016/J.JBI.2023.104580.

[17] A. Ermolina and V. Tiberius, "Voice-controlled intelligent personal assistants in health care: International delphi study," *Journal of Medical Internet Research*, vol. 23, no. 4, p. e25312, Apr. 2021, doi: 10.2196/25312.

[18] S. Z. Kamoonpuri and A. Sengar, "Hi, May AI help you? An analysis of the barriers impeding the implementation and use of artificial intelligence-enabled virtual assistants in retail," *Journal of Retailing and Consumer Services*, vol. 72, p. 103258, May 2023, doi: 10.1016/J.JRETCONSER.2023.103258.

[19] S. Es, J. James, L. Espinosa-Anke, S. Schockaert, and E. Gradients, "Ragas: Automated Evaluation of Retrieval Augmented Generation," Apr. 2025, Accessed: Jul. 04, 2025. [Online]. Available: https://arxiv.org/pdf/2309.15217v2

[20] C. Bérubé *et al.*, "Proactive behavior in voice assistants: A systematic review and conceptual model," *Computers in Human Behavior Reports*, vol. 14, p. 100411, May 2024, doi: 10.1016/J.CHBR.2024.100411.

[21] T. Iqbal *et al.*, "Towards integration of artificial intelligence into medical devices as a real-time recommender system for personalised healthcare: State-of-the-art and future prospects," *Health Sciences Review*, vol. 10, p. 100150, Mar. 2024, doi: 10.1016/J.HSR.2024.100150.

[22] "LangChain." Accessed: Jul. 29, 2025. [Online]. Available: https://www.langchain.com/langchain

[23] “jsvine/pdfplumber: Plumb a PDF for detailed information about each char, rectangle, line, et cetera — and easily extract text and tables.” Accessed: Jul. 29, 2025. [Online]. Available: https://github.com/jsvine/pdfplumber

[24] “Chroma | 🦜🔗 LangChain.” Accessed: Jul. 29, 2025. [Online]. Available: https://python.langchain.com/docs/integrations/vectorstores/chroma/

[25] “ElWilly9/AsistentedeMotosRAG: Asistente Virtual Interactivo para resolución de inquietudes acerca de tu moto implementando RAG.” Accessed: Jul. 29, 2025. [Online]. Available: https://github.com/ElWilly9/AsistentedeMotosRAG

[26] “Quick Introduction | DeepEval - The Open-Source LLM Evaluation Framework.” Accessed: Jul. 29, 2025. [Online]. Available: https://deepeval.com/docs/getting-started