



Implementación de un Chatbot para Recursos Humanos en la Industria Aérea utilizando GenAI y RAG Multimodal

Adolfo Mauricio Órdenes Silva

+56974331715

Adolfo.ordenes@usm.cl

Resumen: La industria aérea de carga y pasajeros se enfrenta a la creciente necesidad de agilidad y eficiencia en la gestión de sus recursos humanos. En este contexto, la presente tesina explora la creación de un chatbot para el departamento de recursos humanos utilizando la tecnología de inteligencia artificial generativa (GenAI) y Recuperación de la Respuesta Asistida (RAG) multimodal. El objetivo es mejorar la comunicación interna, optimizar la gestión de consultas y simplificar los procesos administrativos en las operaciones de recursos humanos de una aerolínea de transporte de carga y pasajeros.

La investigación se centra en cómo el uso de GenAI combinado con RAG multimodal puede proporcionar respuestas precisas y contextualizadas a preguntas comunes de los empleados, mejorando la eficiencia del equipo de recursos humanos y aumentando la satisfacción del personal de la compañía. Se presenta un marco de desarrollo y aplicación del chatbot, destacando la personalización de los modelos de IA para cubrir las necesidades específicas de la industria aérea, así como la integración con sistemas internos para automatizar y facilitar tareas repetitivas.

Además, la implementación de este chatbot impacta directamente en la experiencia de los empleados, ya que mejora la rapidez y precisión en la resolución de sus consultas, lo que se traduce en una mayor satisfacción y una comunicación interna más fluida. La solución optimiza la gestión de recursos humanos al automatizar procesos y facilitar la toma de decisiones en tiempo real. En consecuencia, se evidencia cómo la integración de GenAI y RAG multimodal transforma de manera tangible la operatividad del aérea, estableciendo un modelo de eficiencia que se extiende a otros sectores con necesidades similares de automatización y gestión inteligente.

Palabras Clave: Chatbot, Recursos Humanos, GenAI, RAG multimodal, LLM, Automatización, Industria Aérea.



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☐ Memoria o trabajo de título; ☒ Tesis de Postgrado;

Título del trabajo: Implementación de un Chatbot para Recursos Humanos en la Industria Aérea utilizando GenAI y RAG Multimodal

Nombre del candidato(a): Adolfo Mauricio Órdenes Silva

Carrera / Grado: Magister en tecnologías de la información

Campus: Santiago Vitacura ; **Departamento:** : Departamento de Informática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Marcello Visconti, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 04/08/2025

Firma: 

Estudiante o Candidato(a):

Fecha: 04/08/2025

Firma: 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

1 Introducción

1.1 Contexto, Motivación y problemática

La creciente complejidad de la gestión de recursos humanos en la industria aérea, debido a la gran cantidad de empleados y la diversidad de roles, ha llevado a una necesidad urgente de soluciones tecnológicas que faciliten la comunicación y la eficiencia. Las consultas frecuentes sobre políticas, procedimientos y servicios de recursos humanos representan una carga significativa para los equipos de RH, disminuyendo su capacidad para dedicarse a tareas estratégicas. En este contexto, la motivación para el desarrollo del chatbot surge de la necesidad de ofrecer una herramienta que no solo automatice la resolución de consultas comunes, sino que también mejore la experiencia del empleado al proporcionar respuestas precisas y rápidas. Además, es fundamental garantizar la seguridad de la información, evitando que los empleados carguen documentación en entornos públicos, lo que asegura la protección de los datos sensibles de la empresa. Esta tecnología puede transformar la manera en que se gestionan las interacciones de recursos humanos, reduciendo la carga operativa y aumentando la satisfacción del personal.

1.2 Definición técnica del problema

Desde un punto de vista técnico, el problema se define como la falta de una herramienta automatizada que integre múltiples fuentes de conocimiento y sea capaz de proporcionar respuestas contextualizadas y precisas a las consultas de los empleados. La solución propuesta se basa en el uso de GenAI [1], que permite la generación de respuestas en lenguaje natural, y RAG multimodal [2], que integra y recupera información relevante de diversas fuentes internas, como bases de conocimiento y sistemas de documentación. Esto requiere una arquitectura capaz de manejar consultas complejas, integrarse con sistemas como Confluence [3], Zendesk [4] y otros repositorios, y garantizar la seguridad de los datos a través de la autenticación y perfilamiento de usuarios. La implementación de esta solución busca crear un entorno en el cual los empleados puedan resolver sus dudas de manera autónoma y rápida, optimizando así los procesos de recursos humanos y mejorando la eficiencia operativa de la aerolínea.

1.3 Estrategia o enfoque de solución adoptado

La propuesta de solución consiste en la implementación de un chatbot para recursos humanos en la industria aérea utilizando Generative AI (GenAI) y generación aumentada por recuperación (RAG) multimodal. Además, se utilizará el framework LangChain [5] para facilitar la integración de los diferentes modelos (LLMs) [6] y componentes de la arquitectura de IA (por ejemplo, DB vectoriales), mejorando la eficiencia en el desarrollo y la implementación del sistema. Este enfoque tiene como objetivo optimizar la eficiencia en la gestión de consultas de empleados, automatizando la resolución de preguntas frecuentes y mejorando la experiencia del usuario al ofrecer respuestas rápidas y precisas.

Para el diseño de la solución se consideran los siguientes objetivos:

- **Integración de GenAI y RAG:** Utilizar tecnologías avanzadas de GenAI para generar respuestas en lenguaje natural y RAG multimodal [2] para recuperar información de diversas fuentes internas, como bases de conocimiento y documentos almacenados en sistemas como Confluence y Zendesk.
- **Personalización del modelo:** Personalizar el chatbot para adaptarse a las particularidades del entorno de recursos humanos de la industria aérea, asegurando que la tecnología pueda cubrir las consultas típicas de los empleados y mejorar su experiencia.
- **Arquitectura segura:** Proponer una arquitectura que asegure la protección de los datos sensibles de la empresa, evitando el almacenamiento o la carga de documentos en entornos públicos.
- **Optimización y mejora continua:** Integrar un proceso de revisión automática y manual para mejorar la precisión del chatbot y ajustar los modelos según el feedback recibido.

1.4 Metodología de trabajo para la solución

Las etapas para abordar el trabajo serán las siguientes:

1. Identificación y Preparación de Datos:

- Recopilación de Datos: Recopilar un conjunto de preguntas frecuentes y documentos relevantes de recursos humanos, provenientes de sistemas internos como Confluence, Zendesk y otros repositorios de información corporativa.
- Vectorización de Datos: Convertir la información recopilada en vectores a través de embeddings generados con modelos como GPT-4 [7] o similares, lo cual permitirá una búsqueda semántica más eficiente y precisa.

2. Desarrollo del Modelo de Chatbot:

- Implementación del RAG (Synthetic Reasoning): Utilizar una arquitectura basada en RAG (Retrieval-Augmented Generation) que incluye la búsqueda semántica en múltiples fuentes de conocimiento de la empresa, como Confluence y Zendesk.
- Integración de la Aplicación Web (RH+): Desarrollar e integrar el chatbot en una aplicación web (dominio corporativo), que permitirá a los empleados interactuar de manera fácil con el sistema. El chatbot procesará las consultas mediante RAG para buscar la información relevante, enriqueciendo la respuesta con contexto adicional antes de enviarla al usuario.
- Uso del framework LangChain: Integrar LangChain como herramienta principal para gestionar el flujo de trabajo entre los componentes de recuperación y generación, asegurando una orquestación eficiente y modular entre los modelos de GenAI y RAG.

3. Entrenamiento y Pruebas del Chatbot:

- Entrenamiento del Modelo: Entrenar el modelo de Generative AI (GenAI) utilizando las bases de conocimiento internas almacenadas en sistemas como Confluence y Service Desk, aplicando técnicas de recuperación de contexto y enriquecimiento de conversación para garantizar respuestas precisas.
- Pruebas y Optimización: Realizar pruebas del chatbot en un entorno controlado, clasificando las respuestas en categorías como correctas, incorrectas o incompletas. Esta fase de prueba se apoyará en un proceso iterativo de clasificación y ajuste para mejorar el rendimiento del modelo.

4. Integración y Seguridad:

- Perfilamiento y Privilegios: Garantizar un sistema de perfilamiento para restringir el acceso a la información según los privilegios del usuario. Las consultas se filtrarán antes de acceder al motor de búsqueda semántica para garantizar que se respeten las políticas de acceso a datos sensibles.
- Seguridad de la Información: Evitar la carga de documentos sensibles en entornos públicos y aplicar autenticación segura en cada interacción para proteger los datos corporativos. Esto implica el uso de una capa de "Corpus Search" para la recuperación segura de información dentro del conocimiento empresarial.

5. Validación y Ajustes:

- Validación de la Arquitectura Completa: Validar el rendimiento del chatbot en producción mediante la integración de logs de feedback y evaluación continua con herramientas como Looker. Esto permitirá el monitoreo en tiempo real del desempeño del sistema y la retroalimentación de los usuarios.
- Ajustes Basados en Retroalimentación: Realizar ajustes continuos en la integración de la arquitectura, especialmente en componentes como el vector store, el motor de búsqueda semántica y los modelos de generación de respuestas.

6. Implementación y Mejora Continua:

- Despliegue en Producción: Implementar el chatbot en el entorno de producción, asegurando la actualización periódica del corpus de conocimiento. Los contenidos se actualizarán a intervalos regulares (por ejemplo, dos veces al día) para mantener la base de conocimiento al día con los cambios en la documentación interna.

- Monitoreo y Optimización Iterativa: Utilizar herramientas de monitoreo para evaluar el desempeño del chatbot a través de métricas como la tasa de respuesta efectiva, el número de sesiones por día y la satisfacción del usuario. La mejora continua se logrará a través de un proceso iterativo que incluye análisis del feedback, talleres con los líderes de contenido y ajustes periódicos del modelo.
7. **Modelos Disponibles para el Razonamiento Semántico:**
- Modelos de GenAI y Semántica: Utilizar diferentes modelos como GPT-4, Gemini y otros modelos disponibles para el razonamiento semántico. Cada uno de estos modelos tiene diferentes capacidades de procesamiento (por ejemplo, solicitudes por minuto) y se seleccionará el más adecuado según la carga de consultas y los requisitos del sistema.

Esta metodología asegura que el chatbot esté bien alineado con los requisitos del negocio y que sea capaz de proporcionar respuestas seguras, precisas y rápidas, mejorando así la eficiencia en la gestión de recursos humanos dentro de la industria aérea. La integración detallada de cada uno de los componentes mencionados asegura que el sistema sea robusto, seguro y adaptable a las necesidades cambiantes del entorno empresarial.

1.5 Estimación del impacto de implementación de la propuesta

Las proyecciones del trabajo realizado podrían tener el siguiente impacto:

La implementación del chatbot para recursos humanos tendrá un impacto significativo en diversos aspectos operativos de la empresa. Se espera una reducción del tiempo de respuesta a las consultas de los empleados en un 40-60%, lo cual permitirá liberar recursos y enfocar al personal de recursos humanos en tareas de mayor valor estratégico. Además, el uso del RAG multimodal [2] proporcionará respuestas más precisas, lo cual se traducirá en un aumento del 30-50% en la satisfacción de los empleados al acceder a la información relevante de manera rápida y eficiente.

Otro impacto importante es la reducción en la carga administrativa, logrando automatizar aproximadamente el 70% de las consultas frecuentes de los empleados. Esto se traducirá en una disminución en la necesidad de recursos dedicados a la atención de estas solicitudes, optimizando los costos operativos. Asimismo, el uso de una arquitectura segura y un perfilamiento robusto contribuirá a mitigar riesgos relacionados con la seguridad de la información, lo cual es crucial en un entorno que maneja datos sensibles del personal.

En términos de mejora continua, la integración de herramientas de monitoreo y evaluación permitirá una adaptación constante del sistema a las necesidades cambiantes de la empresa, garantizando un desempeño óptimo a lo largo del tiempo. Se estima que la eficiencia en la gestión del conocimiento se incrementará en un 25-40%, favoreciendo la agilidad y efectividad de los procesos de recursos humanos.

1.6 Hipótesis y método de validación

1.6.1 Hipótesis

La hipótesis que se plantea en el desarrollo de esta tesina es la siguiente:

- "La implementación de un chatbot para el departamento de recursos humanos utilizando GenAI y RAG multimodal [2] permitirá mejorar la eficiencia en la gestión de consultas de los empleados y aumentar la satisfacción del personal en la industria aérea."



1.6.2 Variables de Investigación

- **Variables Independientes:**
 - Modelo de Generative AI (GPT-4) y RAG multimodal [2]. (alguna característica del modelo)
- **Variables Dependientes:**
 - Tasa de satisfacción de los empleados con respecto al acceso a la información.
 - Tiempo promedio de respuesta del chatbot.
 - Reducción de la carga operativa del departamento de recursos humanos.

1.6.3 Método de Validación

- **Desarrollo de Prototipo:** Se desarrollará un prototipo del chatbot utilizando Generative AI y RAG multimodal [2] para pruebas en un entorno controlado.
- **Evaluación de Desempeño:** Se evaluarán las métricas de desempeño del chatbot en términos de precisión, tiempo de respuesta y satisfacción del usuario.
- **Métricas de Validación:** Se utilizarán indicadores como la precisión del modelo y el tiempo de respuesta para evaluar el rendimiento. Adicionalmente, se llevarán a cabo encuestas de satisfacción para medir la percepción de los empleados sobre el sistema.
- **Validación Comparativa:** Comparar los resultados del chatbot con métodos tradicionales de gestión de recursos humanos para determinar el nivel de mejora alcanzado en términos de eficiencia y satisfacción.
- **Validación Cruzada:** Realizar validación cruzada de las métricas mediante la aplicación de diferentes conjuntos de pruebas y obtener una evaluación confiable de la efectividad del chatbot.
- **Pruebas de Escalabilidad:** Evaluar la capacidad del sistema para manejar un volumen creciente de consultas y la efectividad de las respuestas en situaciones de alta demanda.

Este método de validación permitirá asegurar que el chatbot cumpla con los objetivos planteados y que su implementación aporte mejoras tangibles en la gestión de recursos humanos en la industria aérea.

1.7 Estructura del informe

En el segundo capítulo se revisa el marco teórico y estado del arte de los conceptos claves que se utilizarán dentro de esta investigación.

En el tercer capítulo, se aborda el desarrollo de la metodología propuesta con los experimentos realizados y los resultados obtenidos. Se presenta una propuesta de visión arquitectónica del sistema considerando la solución propuesta para dar respuesta a servicios de alta demanda.

Finalmente, se presenta una conclusión respecto de los resultados obtenidos, proponiendo consideraciones y próximos pasos a seguir.

2 Marco teórico y Estado del arte

Las bases de conocimiento de este trabajo se fundamentan en el *Attention Is All You Need* [7], *Improving Language Understanding by Generative Pre-Training* [8], *Retrieval-Augmented Generation for Large Language Models: A Survey* [2] y el *Cloud Computing* [9].

El marco teórico de esta tesina se basa en los avances más recientes en el campo de la inteligencia artificial, particularmente en la utilización de modelos generativos y técnicas de atención para la creación de chatbots de alto rendimiento. Dos de los artículos más influyentes en este contexto son "Attention Is All You Need" y "Improving Language Understanding by Generative Pre-Training". Ambos trabajos han sido fundamentales para el desarrollo de arquitecturas que permiten una mejor comprensión y generación de lenguaje natural en aplicaciones como el chatbot propuesto para recursos humanos en la industria aérea.

2.1.1 Atención y Transformadores: "Attention Is All You Need"

El artículo "Attention Is All You Need" (Vaswani et al., 2017) revolucionó el campo del procesamiento del lenguaje natural (NLP) al introducir el modelo *Transformer*¹, que se ha convertido en la base de numerosos avances en la inteligencia artificial. El concepto de atención permite a los modelos enfocar su capacidad en las partes más relevantes de una secuencia de entrada, lo cual es particularmente útil para manejar tareas complejas de lenguaje.

El modelo *Transformer* reemplazó las tradicionales arquitecturas basadas en recurrencias y convoluciones, proporcionando una estructura más paralelizable y eficiente. El uso del mecanismo de auto-atención y la estructura sin recurrencias permitieron superar problemas como la dependencia a largo plazo y la falta de eficiencia en el entrenamiento de grandes volúmenes de datos. En el contexto del chatbot para recursos humanos, el uso del *Transformer* facilita la comprensión de consultas complejas realizadas por los empleados, y permite al sistema proporcionar respuestas coherentes y relevantes al extraer información de múltiples fuentes.

2.1.2 Scaled Dot-Product Attention

Esta es la fórmula base del mecanismo de atención del *Transformer*. Permite calcular la importancia de cada elemento de la secuencia de entrada con respecto a otros elementos [7].

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Q (queries), K (keys), y V (values) son matrices que contienen la información de entrada. Estas matrices se obtienen aplicando proyecciones lineales a la secuencia de entrada.
- QK^T : Se calcula el producto punto entre las consultas (Q) y las claves (K), resultando en una matriz que representa la similitud entre cada par de elementos de la secuencia.
- d_k : Es un valor de escalado. Se divide el producto punto entre la raíz cuadrada de la dimensión de los vectores clave (d_k) para evitar que los valores del producto punto sean demasiado grandes, lo que podría llevar a la saturación de la función **softmax** y dificultar el aprendizaje.
- **softmax**: Esta función convierte los valores del producto punto en probabilidades, de modo que cada fila representa una distribución de probabilidad sobre las diferentes posiciones de la secuencia.

¹ Un transformador es un modelo de aprendizaje profundo que usa el mecanismo de autoatención, en el que se le da un peso diferente a cada parte de la entrada. Se usa en especial en el campo del procesamiento automático del lenguaje (PAL) y visión artificial. ([https://es.wikipedia.org/wiki/Transformador_\(modelo_de_aprendizaje_autom%C3%A1tico\)](https://es.wikipedia.org/wiki/Transformador_(modelo_de_aprendizaje_autom%C3%A1tico)))

- **V**: Finalmente, se multiplica la salida de la función softmax por los valores (**V**), generando una representación ponderada.

2.1.3 Multi-Head Attention

La atención Multi-Head es una extensión de la atención escalada. Se aplica el mecanismo de atención varias veces (usando diferentes parámetros) para capturar diversas subrepresentaciones [7]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

- **head_i**: Cada cabeza de atención aplica la atención escalada con diferentes matrices de proyección.
- **W_i^Q, W_i^K, W_i^V** : Son matrices de proyección aprendidas que proyectan las consultas, claves y valores en diferentes subespacios antes de aplicar la atención.
- **Concat**: Los resultados de todas las cabezas de atención se concatenan y se proyectan a través de una matriz **W^O** para obtener la salida final.

Este mecanismo permite al modelo aprender diferentes relaciones y patrones dentro de la secuencia.

2.1.4 Feed-Forward Networks (FFN)

Cada capa del Transformer también contiene una red completamente conectada que se aplica a cada posición de manera independiente [7]:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- **W_1, W_2** son matrices de pesos, y **b_1, b_2** son los sesgos.
- **ReLU ($\max(0, \cdot)$)** introduce no linealidad en la red.

La red está compuesta por dos capas totalmente conectadas separadas por una activación **ReLU**. La dimensión de la capa interna suele ser mayor (por ejemplo, 2048) comparada con la dimensión de entrada y salida (generalmente 512).

2.1.5 Positional Encoding

Como el Transformer no tiene una estructura recurrente, necesita una forma de introducir información sobre el orden de las palabras en la secuencia. Esto se hace mediante una codificación posicional sinusoidal[7]:

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

$$\text{PE}(\text{pos}, 2i + 1) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

- **pos**: Es la posición en la secuencia.
- **i**: Es el índice de la dimensión de la codificación.
- **d_{model}** : Es la dimensión del vector de entrada (por ejemplo, 512).

Estas funciones permiten al modelo aprender la posición relativa de los elementos en la secuencia, lo cual es fundamental para capturar la estructura del texto.

Resumen:

- **Scaled Dot-Product Attention:** Calcula una combinación ponderada de los valores (V) en función de la similitud entre las consultas (Q) y las claves (K).
- **Multi-Head Attention:** Utiliza múltiples cabezas de atención para captar diferentes relaciones dentro de la secuencia.
- **Feed-Forward Network (FFN):** Añade no linealidad y capacidad de representación mediante una red completamente conectada.
- **Positional Encoding:** Introduce información posicional en la secuencia, permitiendo al modelo distinguir las posiciones relativas de los elementos.

La combinación de estos elementos permite al Transformer procesar secuencias de manera eficiente y paralela, superando las limitaciones de las redes recurrentes y convolucionales.

2.2 Pre Entrenamiento Generativo: "Improving Language Understanding by Generative Pre-Training"

El segundo artículo clave es "*Improving Language Understanding by Generative Pre-Training*" (Radford et al., 2018) [8]. Este trabajo introdujo el concepto de preentrenamiento generativo (GPT), donde el modelo es primero entrenado en un gran corpus de texto de manera no supervisada, y posteriormente ajustado para tareas específicas mediante aprendizaje supervisado. Esta metodología ha demostrado ser altamente eficaz para mejorar la capacidad de comprensión y generación del lenguaje natural.

El enfoque de preentrenamiento generativo permite que el chatbot tenga un entendimiento profundo del lenguaje humano, lo cual es fundamental para generar respuestas naturales y útiles en el contexto de recursos humanos. El modelo se entrena primero en una amplia variedad de textos generales, lo cual le otorga un conocimiento base extensivo. Luego, se ajusta específicamente a los documentos y políticas internas de la aerolínea, lo cual permite personalizar las respuestas del chatbot y asegurar su relevancia en el entorno de recursos humanos.

Este modelo se basa en varias formulaciones matemáticas fundamentales que permiten comprender cómo se desarrolla el preentrenamiento generativo:

2.2.1 Objetivo de modelado de lenguaje no supervisado

Esta fórmula representa el objetivo de preentrenamiento para el modelo de lenguaje. Se busca maximizar la probabilidad logarítmica condicional de predecir el token u_i , dado un contexto de k tokens anteriores.²

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta)$$

- U : Conjunto no supervisado de tokens
- u_i : El token actual que se intenta predecir.
- k : Tamaño de la ventana de contexto.
- Θ : Parámetros del modelo, que se ajustan durante el entrenamiento.

² Improving Language Understanding by Generative Pre-Training. pp 3 – June 1, 2018

- **Relevancia:** Establece cómo el modelo aprende representaciones a partir de datos de texto no etiquetados.

2.2.2 Arquitectura del modelo Transformer

Describe el flujo de datos a través de las capas del **Transformer** utilizado para el modelado del lenguaje.³

$$h_0 = UW_e + W_p$$

$$h_l = \text{transformer_block}(h_{l-1}) \quad \forall i \in [1, n]$$

$$P(u) = \text{softmax}(h_n W_e^T)$$

- **W_e :** Matriz de embeddings para los tokens.
- **W_p :** Matriz de embeddings de posición, que permite al modelo incorporar información de la posición de cada token.
- **h_l :** Representación en la capa l del Transformer.
- **n :** Número total de capas en el modelo.
- **transformer_block:** Bloque estándar de Transformer que aplica atención multi-head y capas feedforward.
- **Relevancia:** Esta arquitectura permite capturar dependencias a largo plazo en el texto.

2.2.3 Probabilidad condicional supervisada

Establece cómo el modelo calcula la probabilidad de la etiqueta “ y ” a partir de las activaciones de la última capa del Transformer.

$$P(y \mid x_1, \dots, x_m) = \text{softmax}(h_m^l W_y)$$

- **h_m^l :** Activación de la última capa del Transformer.
- **W_y :** Pesos de la capa de salida lineal añadida.
- **Relevancia:** Es esencial para la etapa de ajuste fino supervisado, donde se predicen las etiquetas de las tareas específicas.

2.2.4 Objetivo de ajuste fino supervisado

Este objetivo supervisado ajusta los parámetros del modelo pre entrenado para adaptarse a un conjunto de datos etiquetados C .

$$L_2(C) = \sum_{(x,y)} \log P(y \mid x_1, \dots, x_m)$$

- **C :** Conjunto de datos etiquetados.
- **(x,y) :** Ejemplo de entrada x y etiqueta correspondiente y .
- **Relevancia:** Permite al modelo realizar predicciones precisas en tareas específicas utilizando los conocimientos adquiridos durante el preentrenamiento.

³ Improving Language Understanding by Generative Pre-Training. pp 3 – June 1, 2018

2.2.5 Objetivo combinado para ajuste fino

Combina los objetivos de ajuste fino supervisado (L_2) y pre entrenamiento no supervisado (L_1), ponderados por un factor λ .⁴

$$L_3(C) = L_2(C) + \lambda \cdot L_1(C)$$

- λ : Factor de ponderación que controla la importancia del objetivo no supervisado durante el ajuste fino.
- **Relevancia:** Mejora la generalización y acelera la convergencia durante el ajuste fino.

Estas fórmulas explican cómo el modelo combina pre entrenamiento no supervisado y ajuste fino supervisado para lograr un desempeño superior en tareas de comprensión del lenguaje. La estructura del *Transformer* es clave para capturar relaciones complejas en el texto, mientras que los objetivos combinados aseguran una transferencia efectiva de conocimientos.

2.3 Retrieval Augmented Generation (RAG)

Problema del cliente: Proporcionar búsqueda semántica para encontrar respuestas a partir de un conjunto de datos propietario y obtener respuestas con resúmenes basados en los datos⁵

- Las respuestas de la búsqueda no deben parafrasear los datos almacenados; deberían responder la pregunta.
- Las respuestas de la búsqueda deben tener un puntero al documento donde se encontró la respuesta capaz de verificar que las respuestas se basan en los datos del cliente.
- Las respuestas no se deben inventar si los datos no se encuentran en los resultados de la búsqueda.
- El conjunto de datos es propiedad del cliente y debe permanecer privado.
- El conjunto de datos crece constantemente (no está obsoleto).

2.3.1 ¿Que es Retrieval Augmented Generation (RAG)?

El problema:

- Los LLM no conocen los datos de propiedad de su empresa o específicos del dominio.
- Los LLM no tienen información en tiempo real.
- A los LLM les resulta difícil proporcionar citas precisas.

La solución:

- Alimentar el contexto relevante del LLM en tiempo real, mediante el uso de un sistema de recuperación de información.

⁴ Improving Language Understanding by Generative Pre-Training. pp 3 – June 1, 2018

⁵ <https://aws.amazon.com/es/what-is/retrieval-augmented-generation/>

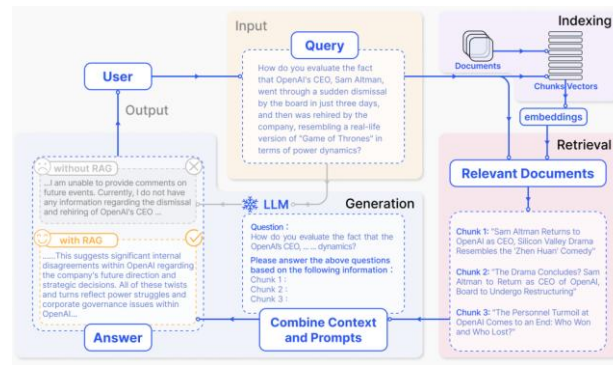


Figura 1: instancia representativa del proceso RAG [9]

Consiste principalmente en 3 pasos:

- **Indexación:** Los documentos se dividen en fragmentos, se codifican en vectores y se almacenan en una base de datos vectorial.
- **Recuperación:** Se recuperan los k fragmentos más relevantes para la pregunta, basándose en la similitud semántica.
- **Generación:** Se introduce la pregunta original y los fragmentos recuperados conjuntamente en el modelo de lenguaje (LLM) para generar la respuesta final.

RAG es una implementación del patrón ReAct (Reasoning and Act).⁶

- Un repositorio de datos externo al LLM.
- Embeddings y búsqueda vectorial para encontrar los datos relevantes para una consulta de usuario
- El LLM responde a la pregunta basándose sobre los datos proporcionados

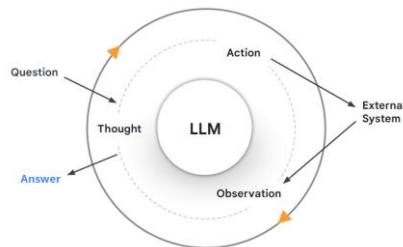


Figura 2: ReAct (Reasoning and Act)

2.4 Trabajos relacionados.

Al revisar diversas investigaciones y estudios relacionados previos para el desarrollo de esta tesina, se encontraron diversos artículos basados RAG como, por ejemplo:

Salud, RAG puede analizar los registros médicos y extraer datos relevantes para ayudar a los médicos en el diagnóstico de condiciones.

Esta tecnología ayuda a los profesionales sanitarios a proporcionar una mejor atención y tomar decisiones más informadas.

⁶ <https://www.promptingguide.ai/techniques/react>

Servicios financieros, en servicios financieros, RAG puede analizar grandes conjuntos de datos para identificar patrones indicativos de actividades fraudulentas. También podría analizar los datos y tendencias del mercado, ayudando a las instituciones financieras a evaluar los riesgos con mayor precisión.

Manufacturing, al analizar los datos de varias etapas del proceso de producción, RAG puede identificar defectos y sugerir mejoras. Esto mejora la calidad del producto y reduce los residuos.

Incluso podría monitorear y ajustar los parámetros de producción en tiempo real para maximizar la eficiencia.

Si necesita ayuda con la gestión de la oferta, RAG AI puede predecir la demanda, optimizar los niveles de inventario y agilizar la logística. Esto reduce los costos y mejora la satisfacción del cliente.

Uno de los trabajos relacionados en materia social, es lo que ha realizado el equipo de Digital Green⁷

Digital Green se asocia con gobiernos y organizaciones en varios países (India, Etiopía, Kenia, Nepal y Nigeria) para mejorar los sistemas de extensión pública agrícola mediante una plataforma impulsada por IA. Esta plataforma apoya a los agentes de extensión (EAs), empleados gubernamentales con experiencia local en agricultura, quienes trabajan directamente con agricultores para aumentar su productividad.

Desafío:

Aunque los EAs son competentes, enfrentan dificultades debido a sistemas ineficaces, carga de trabajo y falta de herramientas digitales. Esto impide la adopción de buenas prácticas agrícolas y afecta la productividad y los ingresos de los agricultores. Con las herramientas de Digital Green, se ha observado un aumento del 20% en los ingresos de los agricultores y un ahorro del 16% en insumos.

Solución:

Digital Green ha desarrollado un asistente digital impulsado por IA (RAG) que ayuda a los EAs a organizar tareas, registrar consultas de agricultores y acceder a información relevante. La aplicación integra contenido científico, meteorológico y financiero, mejorando las recomendaciones a los agricultores. La plataforma también facilita la comunicación bidireccional entre los EAs y el motor de IA de Digital Green, permitiendo que el sistema aprenda y se actualice en tiempo real.

Objetivo Final:

Digital Green espera eventualmente transferir su plataforma a los gobiernos socios, para que se convierta en una herramienta pública de extensión agrícola.



Figura 3: Teck Stack – Digital Green [10]

⁷ <https://digitalgreen.org/ourwork/>

3.1 Desarrollo Implementación de un Chatbot para Recursos Humanos en la Industria Aérea utilizando GenAI y RAG Multimodal

La solución propuesta sigue una arquitectura Cloud Native basada en tecnologías de Google Cloud Platform [12], según se describe a continuación:

- **Cloud Run [13] y FastAPI [14]:** La implementación del Backend del Chatbot se realiza mediante Cloud Run, utilizando FastAPI para manejar las solicitudes del Chatbot, asegurando así una infraestructura escalable que se adapta a las demandas de consultas de los empleados.
- **Cloud Scheduler [15] y BigQuery [16]:** Se utiliza Cloud Scheduler para tareas de actualización periódica y BigQuery para el análisis de métricas y monitoreo, evaluando el rendimiento del Chatbot y realizando ajustes según sea necesario.
- **API Gateway[17] y Load Balancers[18]:** Los usuarios acceden al Chatbot a través de un API Gateway detrás de balanceadores de carga, lo cual asegura alta disponibilidad y una distribución eficiente de la carga de trabajo.
- **Google Cloud SQL[19] y PgVector[20]:** Cloud SQL con PgVector se usa para la gestión de Embeddings, facilitando la búsqueda semántica eficiente y mejorando la precisión de las respuestas.
- **Akamai WAF[21]:** Un Firewall de Aplicaciones Web (WAF) de Akamai se emplea para proporcionar seguridad adicional, protegiendo la infraestructura contra ataques comunes y asegurando la integridad de la información procesada.

Para ello la arquitectura Cloud utilizará las siguientes componentes según la imagen a continuación:

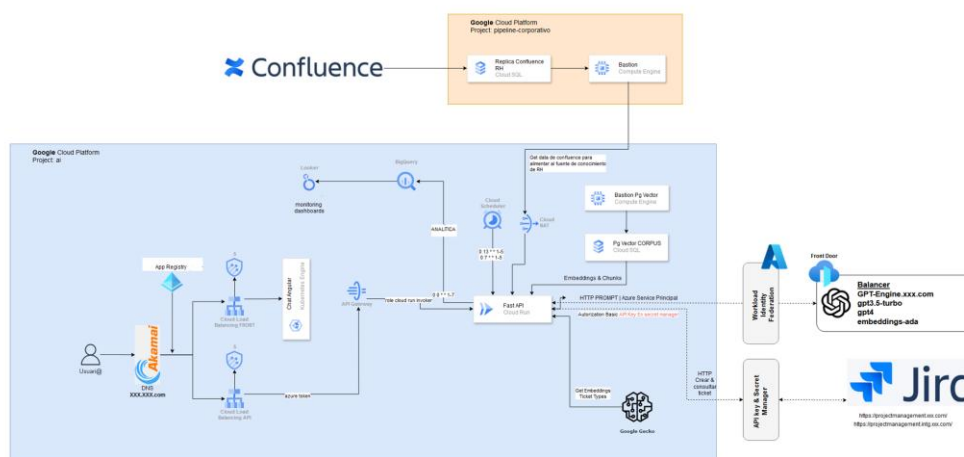


Figura 4: Diagrama de Arquitectura Cloud Chatbot.

[illegible]

14



Figura 6: Confluence, para la documentación de las bases de conocimiento.

Un proceso como el siguiente permite calcular un score de especificidad, en donde por cada documento, se autogeneran preguntas con un LLM que se pasan por el motor de búsqueda para luego verificar si el documento original se encuentra dentro de los documentos encontrados.

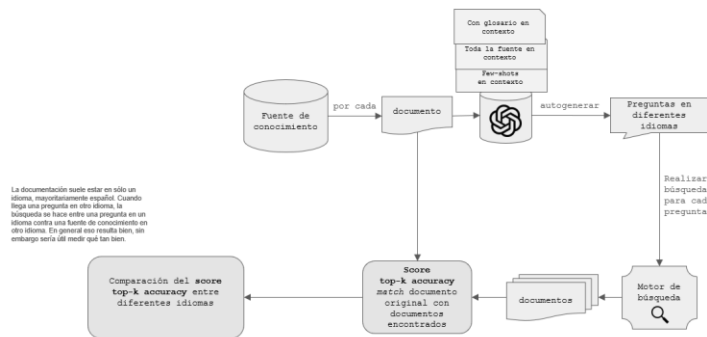


Figura 7: Diagrama de proceso, para la obtención de los documentos.

3.1.2 GPT4 y Framework RAG

3.1.2.1 Modelo de Lenguaje GPT-4 (LLM GPT-4)

El modelo GPT-4 de OpenAI es una de las variables independientes clave en esta investigación. Se trata de un modelo de lenguaje de gran escala basado en la arquitectura Transformer, entrenado para predecir tokens en secuencias de texto. GPT-4 se distingue por su capacidad multimodal, permitiendo la entrada de texto e imágenes y generando texto como salida. En el contexto del chatbot de RRHH en la industria aérea, GPT-4 actúa como el motor generador de respuestas, proporcionando asistencia en consultas de empleados sobre procesos internos, normativas y beneficios laborales. La inclusión de GPT-4 en este estudio se justifica por su alto rendimiento en benchmarks de NLP, su capacidad de personalización mediante ingeniería de prompts y su integración con técnicas de RAG para mejorar la precisión en la recuperación de información.

No obstante, el modelo presenta limitaciones, como la posibilidad de generar respuestas incorrectas (alucinaciones), la existencia de sesgos en los datos de entrenamiento y restricciones en la ventana de contexto. Para mitigar estos factores, se propone una integración con mecanismos de validación y retroalimentación de usuarios. La efectividad del chatbot será evaluada en función de métricas de precisión, coherencia y satisfacción del usuario, considerando el impacto de GPT-4 en la calidad de las respuestas generadas.[9]

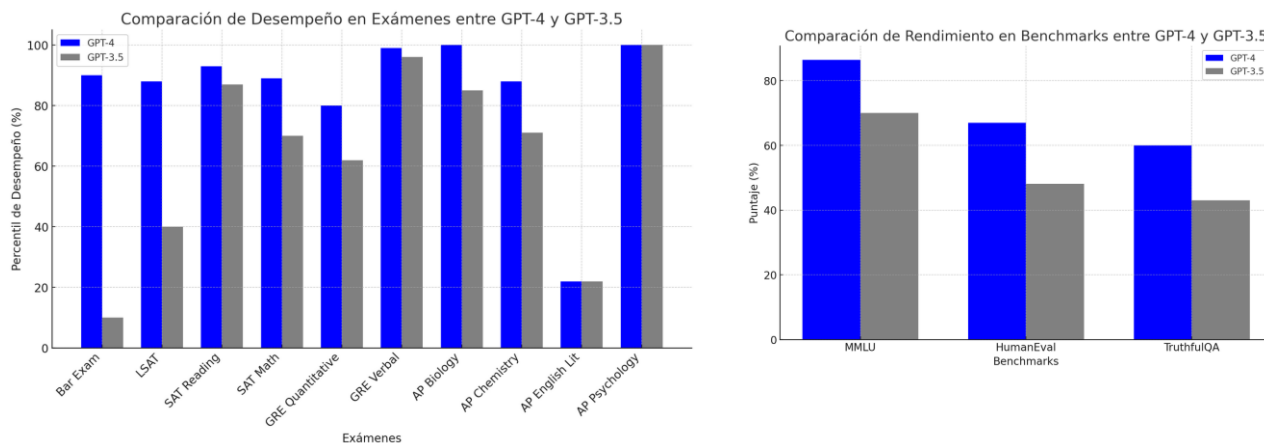


Figura 8: Comparación de desempeño entre GPT-4 y GPT-3.5.[9]

En el primer gráfico representa el desempeño de GPT-4 y GPT-3.5 en distintos exámenes académicos y profesionales, basado en los datos del GPT-4 Technical Report. Se puede observar que GPT-4 supera ampliamente a GPT-3.5, alcanzando percentiles significativamente más altos en pruebas como el **Bar Exam**, **GRE Verbal** y **AP Psychology**.

Para el segundo gráfico, se tiene:

- **MMLU**⁸: GPT-4 alcanza un 86.4%, superando ampliamente el 70% de GPT-3.5.
- **HumanEval**⁹: GPT-4 logra un 67%, mostrando una mejora significativa respecto al 48.1% de GPT-3.5.
- **TruthfulQA**¹⁰: GPT-4 alcanza un 60%, demostrando mayor precisión en la veracidad de respuestas frente al 43% de GPT-3.5.

3.1.2.2 Retrieval-Augmented Generation (RAG)

RAG es una de las variables independientes clave en esta investigación. Se trata de una metodología que combina modelos de lenguaje grande con recuperación de información externa, permitiendo mejorar la precisión y confiabilidad de las respuestas generadas por el chatbot de RRHH.

En este estudio, RAG se implementará en su versión avanzada (**Advanced RAG**), permitiendo la optimización de la indexación, recuperación y filtrado de datos corporativos relevantes para las consultas de los empleados de la aerolínea. La inclusión de RAG se justifica por su capacidad de **mitigar alucinaciones**, mejorar la trazabilidad de respuestas y permitir el acceso a **información actualizada sin necesidad de reentrenamiento del modelo**. No obstante, la efectividad de RAG dependerá de la calidad de los datos recuperados, la optimización del *query* de búsqueda y la latencia en la recuperación de información.

En el marco de esta investigación, se evaluará el impacto de RAG en términos de **precisión de respuesta**, **reducción de alucinaciones** y **satisfacción del usuario**, con el fin de validar su efectividad en el contexto de la industria aérea.

⁸ MMLU es un benchmark diseñado para medir el conocimiento general y la comprensión del lenguaje en una amplia variedad de áreas, incluyendo matemáticas, historia, derecho, biología y más. ref: <https://en.wikipedia.org/wiki/MMLU>

⁹ HumanEval es un benchmark diseñado para evaluar la capacidad de los modelos de lenguaje en **generación de código**. ref: <https://metaschool.so/articles/humaneval-benchmark-for-llm-code-generation>

¹⁰ TruthfulQA es un benchmark diseñado para medir qué tan **veraz** es un modelo de lenguaje, evitando sesgos y **alucinaciones** (respuestas falsas pero convincentes). ref: <https://openai.com/index/truthfulqa/>

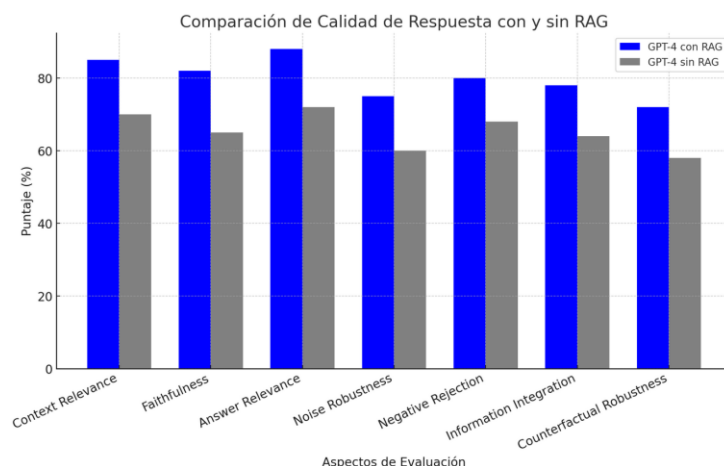


Figura 9: Comparación calidad de respuesta con y sin RAG[2].

3.1.3 Transform y Embeddings

Para el correcto funcionamiento del proceso anterior es necesario aplicar *Transform*[22] y *Embeddings*[23]. Estos son generados usando ML Models, que están entrenados con un enorme corpus de datos. Lo extraordinario de estos modelos que pueden encontrar similitudes en palabras y convertir los textos en vectores.

Un Embedding Vector es una matriz de números que captura el significado semántico de las palabras que representan, por ejemplo:

```
import vertexai
from vertexai.language_models import TextEmbeddingModel

embedding_model = TextEmbeddingModel.from_pretrained("text-embedding-004")

embeddings = embedding_model.get_embeddings(["Python"])

vector = embeddings[0].values
print(f"Length = {len(vector)}")
print(vector)
```

Length = 768
[0.004732487163828611, -0.0061294399201869965, 0.009944838471710682, 0.00749958585947752]

```
embeddings = embedding_model.get_embeddings(["Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models. These vector representations are designed to capture the semantic meaning and context of the words they represent."])

for embedding in embeddings:
    vector = embedding.values
    print(vector)
```

[-0.011677264235913754, -0.04742889106273651, -0.05371042340993881, 0.006055985111743212, -0.01956889125848664]

Figura 10: Ejemplos en Python conversión de texto en vectores.

Para este caso, el siguiente paso en el proceso de recuperación en RAG es posible que tengamos que dividir todo el documento en *chunks* (o partes), que puede caber en esta ventana de contexto de modelo. Aquí es donde los divisores de texto del framework de LangChain entran en acción en la imagen.

```
# Función para dividir documentos en chunks
def split_into_chunks(documents, chunk_size=1000, overlap=200):
    """
    Divide los documentos en chunks manejables.
    """
    splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=overlap)
    chunks = []
    for doc in documents:
        content_chunks = splitter.split_text(doc["content"])
        for chunk in content_chunks:
            chunks.append({"title": doc["title"], "content": chunk})
    return chunks
```

Figura 11: Ejemplo en Python - Recursive chunking.

Lo que se ve en el código anterior representa una de las tantas técnicas para poder generar los chunk llamado “*Recursive chunking*” y se caracteriza por dividir cada trozo en trozos más pequeños, si el tamaño excede un límite de tamaño de trozos predefinido no se hace más la división.

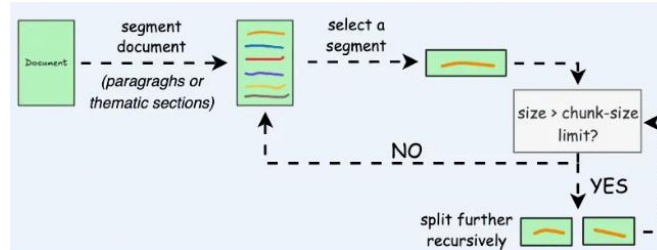


Figura 12: Proceso Recursive chunking [11].

Los **Embeddings** se utilizan para crear una representación vectorial del texto. Estos se almacenan junto con su texto correspondiente en la base de datos de vectores. Usamos una función de inserción para crear incrustaciones de los documentos.

Para este caso utilizamos los modelos de embedding de OpenAI (*text-embedding-ada-002* [24]). Este modelo es uno de los últimos modelos de OpenAI para generar embeddings y rápidamente se ha convertido en una opción destacada para tareas que requieren comprensión semántica. Transforma texto en incrustaciones que se pueden utilizar para aplicaciones posteriores, como búsqueda, recomendación y agrupamiento. Estas son algunas de las características clave:

- Incrustaciones de 1536 dimensiones: *text-embedding-ada-002* genera vectores de 1536 dimensiones, que capturan información semántica rica y detallada.
- Rendimiento de última generación: el modelo funciona excepcionalmente bien en diversas tareas de PNL, proporcionando una comprensión profunda en una amplia gama de entradas de texto.
- Rentable: a pesar de su potencia, *text-embedding-ada-002* está diseñado para ser eficiente, lo que lo hace accesible incluso para aplicaciones con recursos limitados.

```
# Función para obtener embeddings usando el modelo text-embedding-ada-002
def obtener_embeddings(textos):
    embeddings = []
    for texto in textos:
        respuesta = openai.Embedding.create(
            input=texto,
            model="text-embedding-ada-002"
        )
        embeddings.append(respuesta['data'][0]['embedding'])
    return embeddings

# Obtener embeddings
embeddings = obtener_embeddings(info_avion)
```

	Texto	Embeddings
0	Fuselaje: Es el cuerpo principal del avión que...	[-0.020068597048521042, 0.0055471076630055904, ...
1	Ala: Superficie principal que proporciona sust...	[-0.005596087779849768, -0.0011826700065284967, ...
2	Motor: Proporciona la propulsión necesaria par...	[-0.016117689352035522, -0.016906149685382843, ...
3	Tren de aterrizaje: Sistema que soporta al avi...	[-0.029551826417446136, -0.01081622764468193, ...
4	Cola: Incluye los estabilizadores verticales y...	[-0.014170374721288681, -0.0015223356895148754, ...

Figura 12: Ejemplo en Python, para obtener los embeddings y el valor en texto de ellos.

3.1.4 Almacenamientos de Vectores (Store)

Las bases de datos vectoriales, son esencialmente bases de datos diseñadas para un tipo específico de formato de datos: embeddings de vectores. Se especializan en almacenar estos embeddings de manera muy eficiente. Las bases de datos de vectores permiten una recuperación rápida basada en búsquedas de similitudes.

Ahora bien, para este caso aparte de tener claro el uso de una base de datos vectoriales, tuvimos que decidir qué tipo de indexación utilizamos con esto podemos mejorar significativamente la velocidad, escalabilidad y eficiencia en las búsquedas de similitud entre vectores, evitando exploraciones completas de la tabla a medida que el conjunto de datos crece.

Existen dos tipos de índices principales:

- **IVFFlat:**
 - Ofrece resultados exactos del vecino más próximo.
 - Estructura el espacio en clusters, acelerando la búsqueda al dirigirse primero a los grupos relevantes.
 - Equilibra precisión y tiempo de búsqueda.
- **HNSW:**
 - Diseñado para resultados aproximados, priorizando la velocidad.
 - Emplea una estructura gráfica para navegar rápidamente entre vectores.
 - Es muy rápido, pero puede omitir ocasionalmente al vecino más cercano exacto.

En resumen, si bien buscamos exactitud necesitamos buscar con mayor rapidez y toleramos cierta aproximación elegimos HNSW, por sobre IVFFlat.

```
# Función para almacenar embeddings en PostgreSQL
def store_embeddings_in_pg(chunks, embedding_model):
    """
    Genera embeddings para los chunks y los almacena en PostgreSQL.
    """
    # Conexión a PostgreSQL
    conn = psycopg2.connect(**PG_CONFIG)
    cursor = conn.cursor()

    # Crear embeddings e insertar en la base de datos
    for chunk in chunks:
        embedding = embedding_model.embed_query(chunk["content"])
        cursor.execute(
            """
            INSERT INTO document_embeddings (content, embedding)
            VALUES (%s, %s)
            """
            , (chunk["content"], embedding),
        )
    conn.commit()
    cursor.close()
    conn.close()
```

Figura 13: Ejemplo en Python, para guardar los embeddings en la base de datos vectorial.

3.1.5 Retrieval

Siguiendo el concepto de la arquitectura RAG, un punto clave de esta arquitectura está en el “*retrieval*”. Una vez que se generen los **embeddings** de esta fuente de información es necesario recuperarla, de forma correcta. He aquí uno de los grandes desafíos de en materia de IA generativa cuando se construyen soluciones de este tipo. ¿Cómo nos aseguramos de extraer los embeddings de forma correcta?, Sí estoy trabajando con vectores ¿Cómo los comparo? La solución a esto, es encontrar el vecino más cercano de un vector para encontrar semánticamente documentos similares (o imágenes, vídeos, etc.), por ejemplo, con vectores bidimensionales, podemos ver a estos dos vectores más similares.

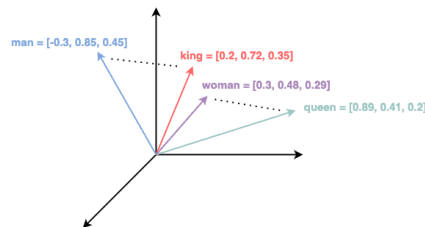


Figura 14: Ejemplo de embedding en el espacio dimensional.

Para eso podemos calcular la distancia entre estos vectores en el espacio de vectores de acuerdo a la métrica de distancia que se adapte mejor a este problema. Entre las más utilizadas tenemos:

L2 Euclidean Distance

La distancia entre dos puntos en un espacio vectorial.

Raíz cuadrada de la suma de diferencias al cuadrado entre las coordenadas correspondientes de dos vectores



Figura 16: Formula Euclidiana con su ejemplo vectorial [26].

L1 Manhattan Distance

Suma de las distancias absolutas entre coordenadas correspondientes de dos vectores.

Menos sensible a los valores atípicos en comparación con el euclidiano

Más rápido de calcular que el euclidiano

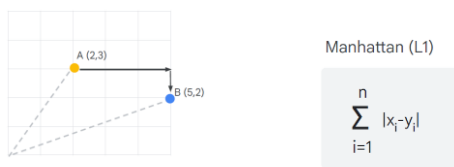


Figura 17: Formula Manhattan con su ejemplo vectorial [26].

Cosine similarity

Devuelve un valor entre -1,0 y 1,0 1.0 significa que los vectores son iguales -1.0 significa que son opuestos entre sí No afectado por la magnitud de los vectores. Alta similitud incluso en textos de diferentes longitudes.

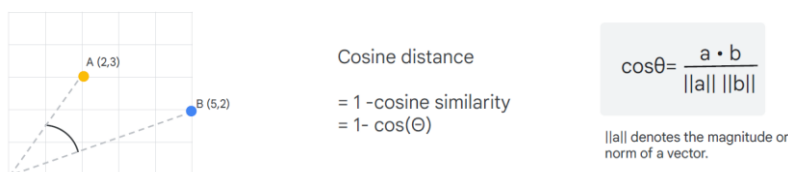


Figura 18: Formula *Cosine similarity* con su ejemplo vectorial [26].

Para este proyecto se utiliza **Cosine similarity**:

```
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.schema import Document
import pandas as pd

def search_golde_set(df, search_term, n=3, pprint=True):
    """
    Busca las notas más similares al término de búsqueda en un DataFrame utilizando LangChain y FAISS.

    Args:
        df (pandas.DataFrame): DataFrame que contiene las notas a buscar.
        search_term (str): El término de búsqueda.
        n (int, optional): La cantidad de resultados a devolver. Por defecto, 3.
        pprint (bool, optional): Si se deben imprimir los resultados. Por defecto, True.

    Returns:
        list[Document]: Lista de documentos más similares al término de búsqueda, ordenados por similitud.
    """
    # Convertir las cadenas en la columna "embedding" a documentos LangChain.
    documents = [
        Document(page_content=row["content"], metadata={"embedding": eval(row["embedding"]), "id": row["id"]})
        for _, row in df.iterrows()
    ]

    # Crear un índice FAISS a partir de los embeddings con similitud coseno.
    embeddings = OpenAIEmbeddings(model="text-embedding-ada-002")
    vectorstore = FAISS.from_documents(documents, embeddings, metric="cosine")

    # Realizar la búsqueda utilizando el término dado.
    results = vectorstore.similarity_search(search_term, k=n)

    if pprint:
        for i, result in enumerate(results):
            print(f"Resultado {i + 1}: \nContenido: {result.page_content} \nMetadata: {result.metadata} \n")

    return results
```

Figura 19: Ejemplo en Python con el uso de **Cosine similarity**, utilizando el framework LangChain.

	question	answer	email	creation_time	http_status	question_embeddings	answer_embeddings	similarities	success
0	mi contraseña ha expirado. Como la renuevo?	Para renovar tu contraseña de cuenta. pu...	@	2023-08-09 13:46:06.333969+00:00	500	[-0.029414648190140724, -0.016450509428977966, ...	[-0.0032174561638385057, -0.005408305674791336, ...	0.798055	True
1	?	Esta opción permite gestionar la entrega de ma...	@	2023-08-09 13:42:33.063603+00:00	500	[-0.004762458149343729, -0.02438824437558651, ...	[0.0021467842161655426, -0.013636372963455658, ...	0.798628	True
2	que hago si expiro mi cuenta de red?	Si tu cuenta de red ha expirado, debes comunic...	@	2023-08-09 13:45:16.115434+00:00	500	[-0.025803069420289993, -0.015317187648451264, ...	[-0.00599377416074276, -0.02421027600215809, 0...	0.817506	True

Figura 20: Resultado de salida basado en la búsqueda de una pregunta.

3.1.6 Front End Chatbot RH

Para poder interactuar con los usuarios, se desarrolló la interfaz del asistente virtual con angular. Este tiene las siguientes características:

Encabezado y nombre de usuario

En la parte superior, nombre del usuario (ejemplo: “Adolfo Mauricio Órdenes Silva”) y el título “Bienvenido a RH”, lo que indica que el usuario está autenticado y se le presenta la sección principal del sistema.

Secciones de consultas rápidas

Justo debajo del título, hay tres tarjetas (o módulos) con temas destacados: **“Travel”**, **“Service Desk”** y **(con diferentes subtítulos como “¿Cómo cambio mi password?” o “Tengo problemas con mi computador”)**. Estas tarjetas funcionarían como accesos directos para preguntas frecuentes o para iniciar conversaciones sobre tópicos específicos.

Lista de temas sugeridos

En la parte central, una lista de secciones o categorías principales (p. ej., **“Teléfono, WiFi, otros”**, **“Workspace”**, **“Tickets y Solicitudes”**, etc.), para guiar al usuario en lo que puede preguntarle al chatbot.

Menú lateral

En el lado izquierdo, se ve un menú con opciones como:

- “Nuevo chat” para iniciar una conversación desde cero.
- “¡Déjanos tu feedback!” para que el usuario evalúe la experiencia.
- “Ayuda” y “Configuraciones” para acceder a ajustes o información adicional.

Selector de idioma

En la parte inferior izquierda se encuentra un desplegable para elegir entre español, inglés y portugués, evidenciando la funcionalidad bilingüe de la aplicación.

Zona de interacción con el chatbot

En la parte inferior, el usuario puede escribir su pregunta o consulta (“Pregúntame lo que quieras...”), acompañado por iconos para enviar el mensaje y reconocimiento de voz.

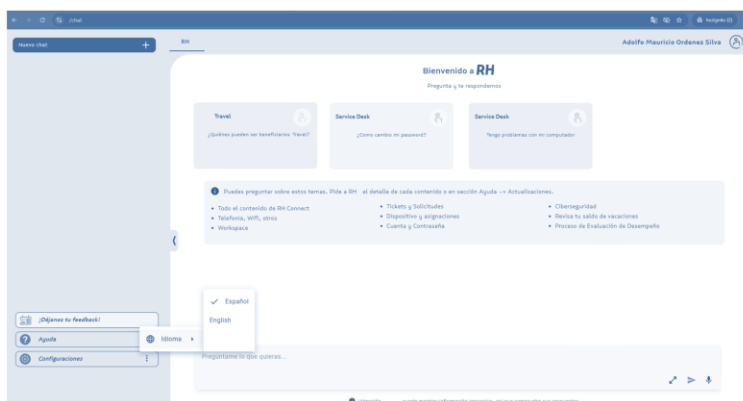


Figura 21: Interfaz del chatbot para los usuarios.

A continuación, se muestran 3 ejemplos interactuando con la plataforma mostrando las respuestas asociadas según base de conocimientos previamente cargadas según su goldenset.

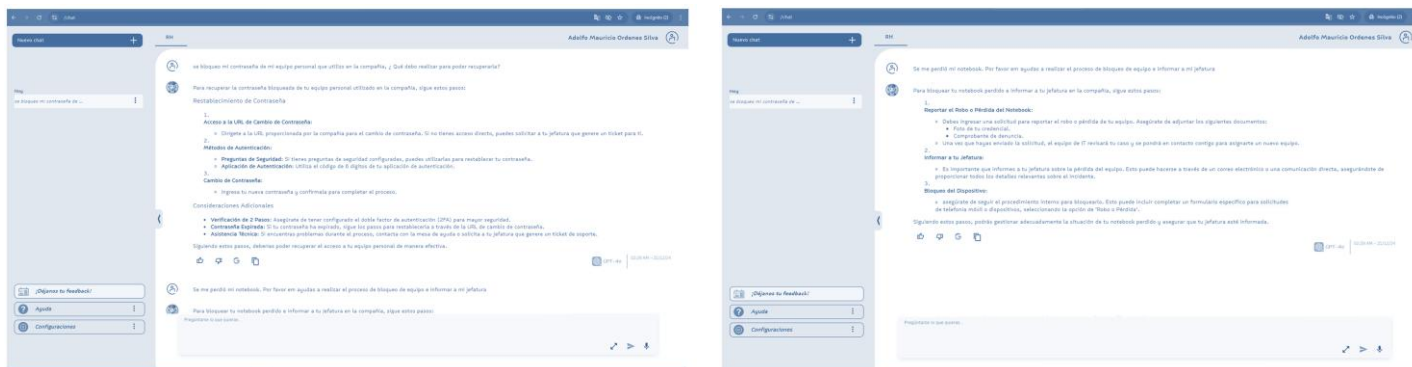


Figura 22: Respuestas del chatbot, al realizar de un par de preguntas por el usuario basadas en seguridad.

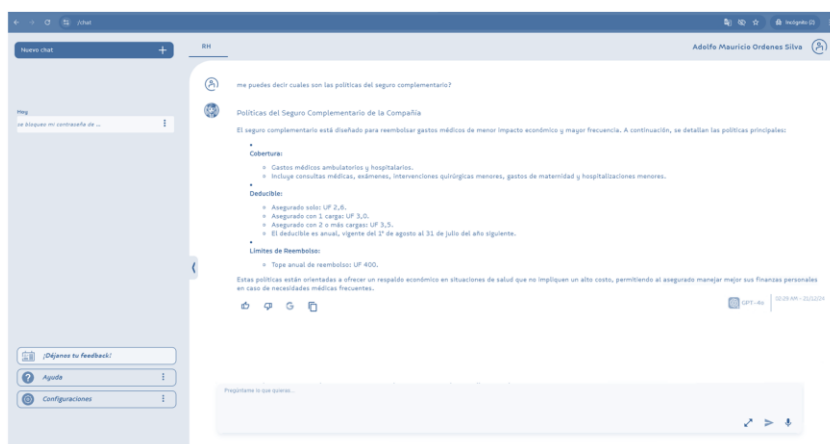


Figura 23: Respuestas del chatbot, al realizar de un par de preguntas por el usuario basadas en seguridad.

3.1.7 Datos y análisis

Para el análisis de resultados respecto al uso del chat y su respectivo análisis de las variables dependientes se toma una muestra de 8.874 usuarios activos de la plataforma, con interacciones mensuales y semanales, con una fecha de inicio del 15 de octubre hasta el 30 de noviembre del año 2024, dado lo anterior se tiene lo siguiente:



Figura 24: Meticas obtenidas del chatbot desde la plataforma Looker Studio.

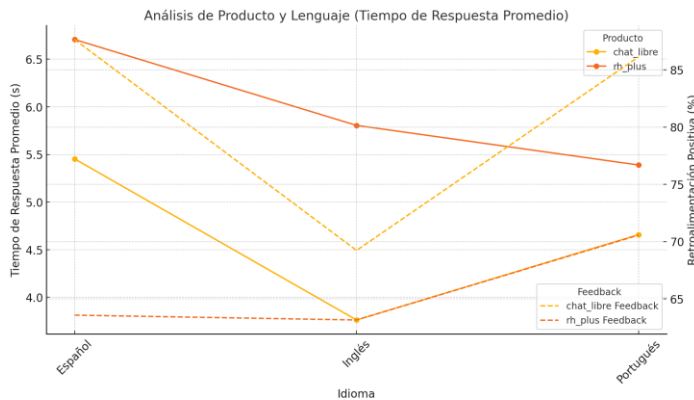
Para el dashboard anterior, se extrae el siguiente dataset “*Data_prod_chatbot*” a analizar según las columnas

Columna	Descripción
Fecha	Marca de tiempo de la interacción.
Producto	Producto específico (por ejemplo, "chat libre", "rh plus").
Idioma	Idioma utilizado (por ejemplo, español, portugués).
Pregunta ID	Identificador único para cada pregunta.
Pregunta	La pregunta del usuario.
Respuesta	La respuesta del chatbot.
Biblioteca	Indica si la respuesta provino de una biblioteca (GoldenSet) predefinida (verdadero/falso).

Contenido	Contenido adicional relacionado con la respuesta (por ejemplo, referencias a documentos).
Categoría	Categoría de la pregunta
Sub-Categoría	Clasificación adicional
Modelo	El modelo utilizado (por ejemplo, "gpt_4o_paygo_global")
Tiempo de respuesta (Segundos)	Tiempo de respuesta en segundos
Calidad de Preguntas	Calidad de la pregunta (por ejemplo, "Buena", "Mala").
Feedback	Retroalimentación del usuario (👍 o 👎).
Comentario feedback	Comentarios adicionales del usuario (si los hay).

Figura 24: Tabla de detalle de las columnas del dataset “*Data_prod_chatbot*”.

Análisis de producto y lenguaje:



Análisis De Producto Y Lenguaje

	Producto	Idioma	avg_response_time	positive_feedback_percentage
1	chat_libre	Español	5.453802281368821	87.642585513308
2	chat_libre	Inglés	3.7615384615384615	69.23076923076923
3	chat_libre	Portugués	4.657651245551602	86.12099644128114
4	rh_plus	Español	6.707444668008048	63.581488933601605
5	rh_plus	Inglés	5.8052631578947365	63.1578947368421
6	rh_plus	Portugués	5.389516129032257	70.56451612903226

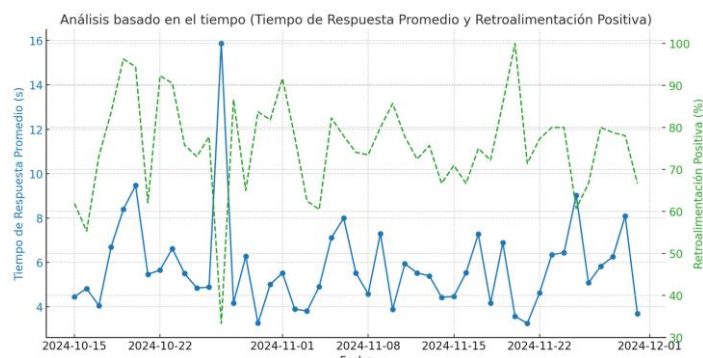
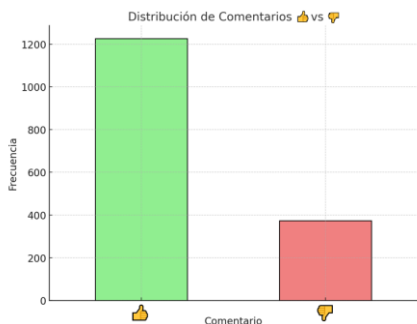
El gráfico muestra cómo varían el **tiempo de respuesta promedio** y el **porcentaje de retroalimentación positiva** en función del **producto** y el **idioma**. Esto nos permite observar si ciertos productos o idiomas están asociados con un mejor rendimiento o mayor satisfacción por parte de los usuarios.

Tasa de satisfacción de los empleados con respecto al acceso a la información:

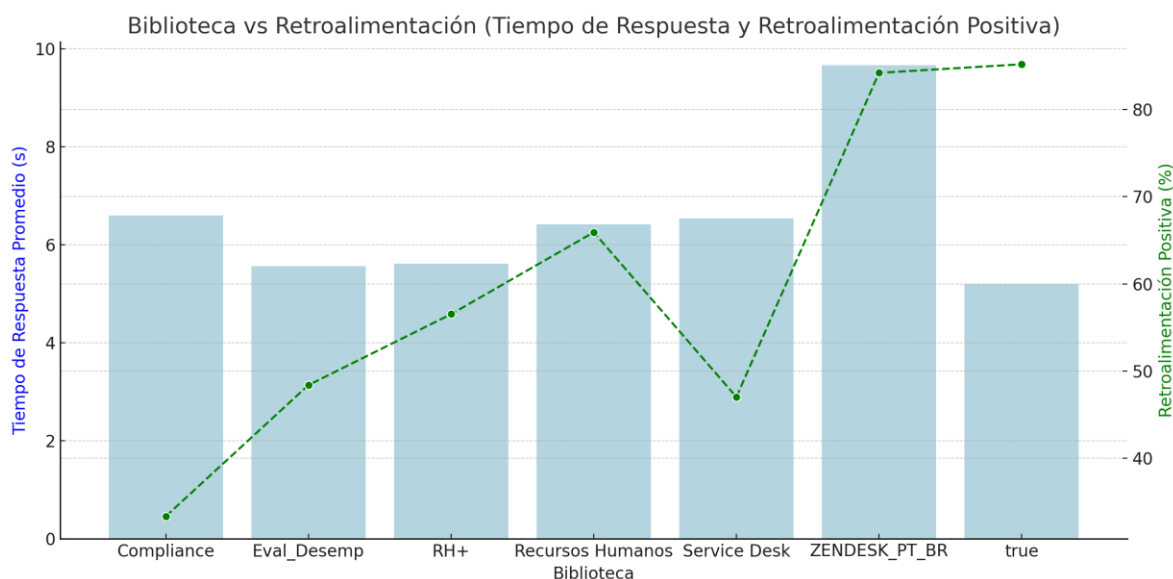
Este indicador podría calcularse a partir del **feedback** de los usuarios. Dado que el **feedback** está representado por los emojis 👍 (positivo) y 👎 (negativo), podemos calcular la **tasa de satisfacción** como el porcentaje de respuestas con feedback positivo.

Fórmula:

$$\text{Tasa de satisfacción} = \frac{\text{Número de respuestas positivas (👍)}}{\text{Total de respuestas}} \times 100$$



Tasa de satisfacción de los empleados con respecto al acceso a la información: **76.63%** (este valor representa el porcentaje de interacciones con feedback positivo).



Las respuestas provenientes de la **biblioteca predefinida** del chatbot (**GoldenSet**) pueden ser percibidas de manera diferente por los usuarios en comparación con las respuestas generadas dinámicamente.

Explicación:

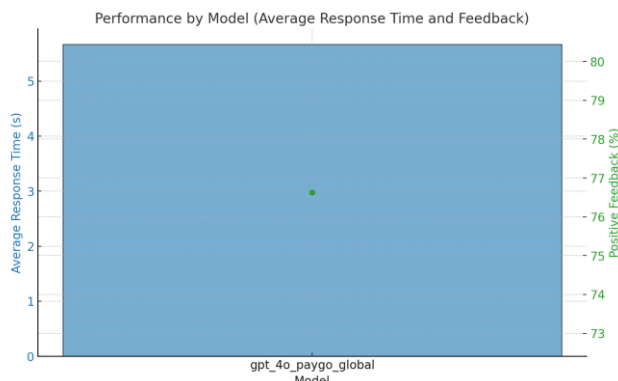
- Las respuestas de la biblioteca pueden ser más rápidas, pero podrían percibirse como menos personalizadas.
- Las respuestas generadas dinámicamente podrían ser más relevantes y específicas, pero a veces podrían tomar más tiempo.

Tiempo promedio de respuesta del chatbot

Este indicador ya se puede calcular directamente utilizando la columna **Tiempo respuesta [s]** del dataset. Simplemente calculamos el **promedio** de los tiempos de respuesta en el dataset.

Fórmula:

$$\text{Tiempo promedio de respuesta} = \frac{\sum \text{Tiempos de respuesta}}{\text{Número total de interacciones}}$$



Model Performance Metrics

	Modelo	avg_response_time	positive_feedback_percentage
1	gpt_4o_paygo_global	5.6655	76.625

Tiempo promedio de respuesta del chatbot: **5.67 segundos** (este es el tiempo promedio que el chatbot tarda en responder a las preguntas).

Cada modelo de inteligencia artificial, como "gpt_4o_paygo_global", puede tener un rendimiento diferente. Compararemos los diferentes modelos en cuanto a los tiempos de respuesta y los comentarios recibidos para identificar cuál es el más eficiente.

Explicación:

- Algunos modelos pueden ser más rápidos, mientras que otros pueden ofrecer respuestas más precisas o relevantes, lo que afecta los comentarios y la satisfacción general de los usuarios.
- Analizar el rendimiento de los modelos ayuda a seleccionar el mejor para futuras interacciones.

Reducción de la carga operativa del departamento de recursos humanos

Este indicador es más subjetivo, ya que necesitaríamos tener datos específicos sobre la carga operativa antes y después de la implementación del chatbot. Sin embargo, como una aproximación, podemos estimar la **reducción de la carga operativa** observando la **cantidad de interacciones automatizadas** que el chatbot maneja en lugar del personal de recursos humanos. Esto podría implicar contar la cantidad de interacciones en categorías relacionadas con recursos humanos (por ejemplo, consultas sobre beneficios, políticas, procesos, etc.), comparándolas con la cantidad de consultas que antes requerirían intervención humana.

Una forma de aproximarlos sería:

- Identificar las interacciones que están relacionadas con el área de **Recursos Humanos** a partir de las **categorías y subcategorías**.
- Calcular el porcentaje de interacciones de recursos humanos que el chatbot ha manejado frente al total de interacciones.

Fórmula aproximada:

$$\text{Reducción de carga operativa} = \frac{\text{Interacciones de Recursos Humanos gestionadas por el chatbot}}{\text{Total de interacciones de Recursos Humanos}} \times 100$$

Si consideramos solo las interacciones gestionadas por "rh", la reducción de carga operativa del departamento de recursos humanos es del 47.81%. Esto indica que aproximadamente la mitad de las interacciones relacionadas con recursos humanos fueron manejadas por el chatbot con esta variable.

4 Conclusión

La implementación de un chatbot para la gestión de recursos humanos en la industria aérea, utilizando GenAI y RAG multimodal, representa un avance significativo en la **optimización de procesos y la mejora de la experiencia del empleado**. A través, de una arquitectura robusta y segura, que integra modelos avanzados de generación y recuperación de información, esta solución demuestra ser eficaz para abordar las necesidades específicas de una industria altamente regulada y compleja.

Los resultados obtenidos reflejan un impacto positivo en tres aspectos clave: la reducción del tiempo de respuesta a consultas frecuentes, la disminución de la carga operativa del equipo de recursos humanos y el aumento de la satisfacción de los empleados. Estas mejoras se lograron gracias a la capacidad del sistema para ofrecer respuestas precisas, contextualizadas y seguras, a la vez que se mantiene adaptable a los cambios dinámicos de la organización.

No obstante, este proyecto no está exento de desafíos. Entre ellos destacan la necesidad de una mejora continua del modelo, la actualización constante del corpus de conocimiento y la gestión efectiva de los feedbacks de los usuarios. Para abordar estos retos, es fundamental establecer un ciclo iterativo de optimización, basado en el monitoreo de métricas clave como la tasa de satisfacción, el tiempo promedio de respuesta y la efectividad en la resolución de consultas.

En un contexto más amplio, este desarrollo pone de manifiesto cómo las tecnologías emergentes como GenAI y RAG multimodal pueden transformar no solo la gestión de recursos humanos, sino también otros procesos críticos en sectores diversos. La replicabilidad de esta solución a otras industrias con necesidades similares de automatización y eficiencia resalta su **valor estratégico**.

Finalmente, este trabajo sienta las bases para futuras investigaciones en la integración de tecnologías generativas con sistemas corporativos. Las áreas de mejora incluyen la exploración de nuevos modelos de IA, exploración de nuevas arquitecturas (como las de **AGENTIC RAG**), el enriquecimiento del chatbot con capacidades de análisis predictivo y la expansión hacia aplicaciones que aborden problemas más complejos en la experiencia del empleado.



REFERENCIAS

- [1] ¿Qué es la IA generativa?: <https://aws.amazon.com/es/what-is/generative-ai/>
- [2] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang, “Retrieval-Augmented Generation for Large Language Models: A Survey”: <https://arxiv.org/abs/2312.10997v5>
- [3] <https://www.atlassian.com/software/confluence>
- [4] <https://www.zendesk.com/#>
- [5] <https://www.langchain.com/langchain>
- [6] LLM: <https://aws.amazon.com/es/what-is/large-language-model/>
- [7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”: <https://arxiv.org/abs/1706.03762>
- [8] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, “Improving Language Understanding by Generative Pre-Training”: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf, pp 3, 2018
<https://openai.com/index/language-unsupervised/>
- [9] OPENAI “GPT-4 Technical Report”: <https://arxiv.org/abs/2303.08774>
- [10] Digital Green: <https://digitalgreen.org/catalyzing-the-data-sharing-ecosystem-introducing-farmstack/>
- [11] Chunking Strategies For RAG: <https://blog.dailydoseofds.com/p/5-chunking-strategies-for-rag?=&s=03>
- [12] Google Cloud: <https://cloud.google.com/docs/overview?hl=es-419>
- [13] Cloud Run: <https://cloud.google.com/run/docs/overview/what-is-cloud-run?hl=es-419>
- [14] FastAPI: <https://fastapi.tiangolo.com/>
- [15] Cloud Scheduler: <https://cloud.google.com/scheduler/docs/overview?hl=es-419>
- [16] BigQuery: <https://cloud.google.com/bigquery/docs/introduction?hl=es-419>
- [17] API Gateway: <https://cloud.google.com/api-gateway/docs/about-api-gateway?hl=es-419>
- [18] Load Balancing: <https://cloud.google.com/load-balancing/docs/load-balancing-overview?hl=es-419>
- [19] Cloud SQL: <https://cloud.google.com/sql/docs/introduction?hl=es-419>
- [20] PgVector: <https://github.com/pgvector/pgvector>
- [21] Akamai WAF: <https://www.akamai.com/es/glossary/what-is-a-waf>
- [22] Transformers: <https://aws.amazon.com/what-is/transformers-in-artificial-intelligence/>
- [23] Embeddings: <https://aws.amazon.com/what-is/embeddings-in-machine-learning/>
- [24] text-embedding-ada-002: <https://openai.com/index/new-and-improved-embedding-model/>
- [25] hns w vs ivfflat: <https://medium.com/@bavalpreetsinghh/pgvector-hns w-vs-ivfflat-a-comprehensive-study-21ce0aaab931>
- [26] Importance of Distance Metrics in Machine Learning Modelling: <https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d>
- [27] <https://cloud.google.com/learn/what-is-cloud-computing?hl=es>

ANEXOS

Análisis Comparativo de GPT-3.5 vs. GPT-4 como Evaluadores de Respuestas.

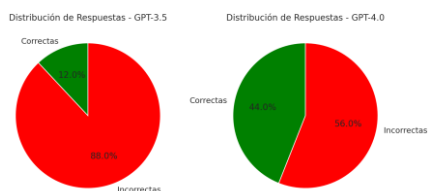
Se analizaron 34 preguntas evaluadas tanto por GPT-3.5 como por GPT-4, desde el dataset *Data_prod_chatbot*. Los modelos fueron calificados en una escala de 0.0 a 1.0, donde 1.0 representa una respuesta perfectamente alineada con el "output esperado". A continuación, se presenta un desglose detallado:

- Tabla de detalle comparativa GPT-3.5 vs GPT-4 del dataset "*Data_prod_chatbot*"

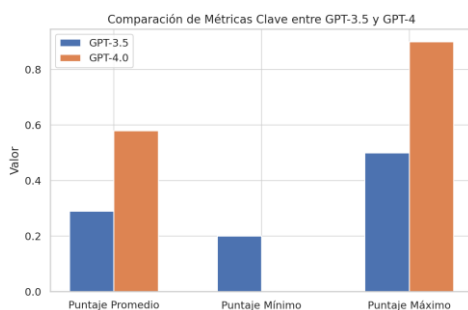
*Correctas = Score ≥ 0.7 (umbral de corte definido en el archivo).

Métrica	GPT-3.5	GPT-4
Puntaje Promedio	0.29	0.58
Puntaje Mínimo	0.2	0.0
Puntaje Máximo	0.5	0.9
Respuestas Correctas*	12%	44%
Respuestas Incorrectas	88%	56%

- Gráfico torta detalle comparativa GPT-3.5 vs GPT-4 del dataset "*Data_prod_chatbot*"



- Distribución de Scores



GPT-3.5: El 70% de sus respuestas obtuvieron **0.2**, lo que indica una tendencia a errores graves (ej: omisión de pasos críticos o información clave).

GPT-4: Mayor variabilidad, con un 35% de respuestas en el rango 0.7-0.9, demostrando capacidad para manejar consultas complejas (ej: "golden_set_87" sobre creación de COBUS mixta).

- Comparación Directa por Pregunta



- En el 82% de los casos, **GPT-4 superó a GPT-3.5**.
- Solo en 2 preguntas ambos modelos tuvieron el mismo desempeño (ej: "golden_set_95" sobre beneficio de luna de miel).