

Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ciencias Matemáticas

Escuela profesional de Computación Científica

Implementación de un asistente virtual basado en inteligencia artificial generativa para la plataforma Vincúlate

TESIS

Para optar el Título Profesional de Licenciado en Computación Científica

AUTOR

Renzo Osmar SALAZAR CARAZAS

ASESOR

Mg. Jhelly Reynaluz PÉREZ NÚÑEZ

Lima, Perú

2024



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Salazar, R. (2024). *Implementación de un asistente virtual basado en inteligencia artificial generativa para la plataforma Vincúlate* [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Escuela Profesional de Computación Científica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Renzo Osmar Salazar Carazas
Tipo de documento de identidad	DNI
Número de documento de identidad	46285148
URL de ORCID	https://orcid.org/0009-0005-2773-8595
Datos de asesor	
Nombres y apellidos	Jhelly Reynaluz Pérez Núñez
Tipo de documento de identidad	DNI
Número de documento de identidad	44076412
URL de ORCID	https://orcid.org/0000-0003-0717-8277
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Edinson Raul Montoro Alegre
Tipo de documento	DNI
Número de documento de identidad	09627181
Miembro del jurado 1	
Nombres y apellidos	Martha Hilda Timoteo Sánchez
Tipo de documento	DNI
Número de documento de identidad	25702279
Miembro del jurado 2	
Nombres y apellidos	Jhelly Reynaluz Pérez Núñez
Tipo de documento	DNI
Número de documento de identidad	44076412
Datos de investigación	
Línea de investigación	No aplica
Grupo de investigación	No aplica
Agencia de financiamiento	Sin financiamiento

Ubicación geográfica de la investigación	Edificio: Universidad Nacional Mayor de San Marcos País: Perú Departamento: Lima Provincia: Lima Distrito: Cercado de Lima Latitud: -12.06041 Longitud: -77.08228
Año o rango de años en que se realizó la investigación	Abril 2024 - Noviembre 2024
URL de disciplinas OCDE	Ciencias de la computación https://purl.org/pe-repo/ocde/ford#1.02.01 Ciencias de la información https://purl.org/pe-repo/ocde/ford#1.02.02

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
FACULTAD DE CIENCIAS MATEMÁTICAS
Escuela Profesional de Computación Científica

ACTA DE SUSTENTACIÓN DE TESIS PARA LA OBTENCIÓN DEL TÍTULO PROFESIONAL DE LICENCIADO EN COMPUTACION CIENTÍFICA (PROGRAMA DE TITULACIÓN PROFESIONAL 2024)

En la UNMSM – Ciudad Universitaria – Facultad de Ciencias Matemáticas, siendo las 10:28 horas del sábado 28 de setiembre del 2024, se reunieron los docentes designados como Miembros del Jurado Evaluador (PROGRAMA DE TITULACIÓN PROFESIONAL 2024): Dr. Edinson Raúl Montoro Alegre (presidente), Dra. Martha Hilda Timoteo Sánchez (miembro) y la Mg. Jhelly Reynaluz Pérez Núñez (miembro asesor), para la sustentación de la Tesis titulada: «IMPLEMENTACIÓN DE UN ASISTENTE VIRTUAL BASADO EN INTELIGENCIA ARTIFICIAL GENERATIVA PARA LA PLATAFORMA VINCÚLATE» presentado por el Bachiller RENZO OSMAR SALAZAR CARAZAS, para optar el Título Profesional de Licenciado en Computación Científica.

Luego de la exposición de la Tesis, el presidente invitó al expositor a dar respuesta a las preguntas formuladas.

Realizada la evaluación correspondiente por los Miembros del Jurado Evaluador, el expositor mereció la aprobación .....bueno....., con un calificativo promedio de 15 (Quince)

A continuación, los miembros del Jurado Evaluador dan manifiesto que el participante bachiller RENZO OSMAR SALAZAR CARAZAS, en vista de haber aprobado la sustentación de su Tesis, será propuesto para que se le otorgue el Título Profesional de Licenciado en Computación Científica.

Siendo las 11:25 horas se levantó la sesión firmando para constancia la presente Acta.

Dr. Edinson Raúl Montoro Alegre
PRESIDENTE

Dra. Martha Hilda Timoteo Sánchez
MIEMBRO

Mg. Jhelly Reynaluz Pérez Núñez
MIEMBRO ASESOR

CERTIFICADO DE SIMILITUD

Yo, Jhelly Reynaluz Pérez Núñez en mi condición de asesor acreditado con la Resolución Decanal N.° 001306-2024-D-FCM/UNMSM de la tesis cuyo título es Implementación de un Asistente Virtual Basado en Inteligencia Artificial Generativa para la Plataforma Vincúlate, presentado por el bachiller Renzo Osmar Salazar Carazas para optar el título profesional de licenciado en Computación Científica. CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante el software de similitud textual, el documento evaluado cuenta con el porcentaje de 19% de similitud, nivel permitido para continuar con los tramites correspondientes y para su publicación en el repositorio institucional.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del título correspondiente.

Firma del asesor:

DNI: 44076412

Nombres y apellidos del asesor: Jhelly Reynaluz Pérez Núñez

Huella digital



Dedicatoria

A mis papitos, Víctor e Isabel, por todo su amor.

A mi mamá María, quien ha desempeñado el papel de madre y padre en mi vida. Su esfuerzo y apoyo incondicionales me han impulsado a superar todas mis metas académicas.

De igual manera, a mi hermana Victoria, tíos y primos, cuya unidad y cariño constantes han sido una fuente de fortaleza para mí.

Agradecimientos

A la Universidad Nacional Mayor de San Marcos, por haberme formado como un buen profesional.

Agradezco a mi asesora Jhelly Pérez por su paciencia, comprensión y guía en el desarrollo de este trabajo.

A los profesores de mi Facultad de Ciencias Matemáticas, por la exigencia y rigurosidad en la formación brindada

ÍNDICE GENERAL

CAPÍTULO I: INTRODUCCIÓN 10

1.1. Situación Problemática 11

1.2. Formulación del Problema 12

1.2.1. Problema General 12

1.2.2. Problema Específicos 12

1.3. Objetivos de la investigación 13

1.3.1. Objetivos General 13

1.3.2. Objetivos Específicos 13

1.4. Justificación 13

CAPÍTULO II: MARCO TEÓRICO 14

2.1. Antecedentes 14

2.2. Bases Teóricas 15

2.2.1. Inteligencia Artificial Generativa 15

2.2.2. Grandes Modelos de Lenguaje 17

2.2.3. Recuperación Aumentada por Generación (RAG) 19

2.2.4. Plataforma Vincúlate 23

2.2.5. Tecnologías de Desarrollo 26

CAPÍTULO III: METODOLOGÍA 29

3.1. Tipo y Diseño de Investigación 29

3.1.1. Unidad de Análisis 29

3.1.2. Población de Estudio 29

3.1.3. Tamaño de Muestra 29

3.1.4. Selección de Muestra 30

3.2. Técnicas de Recolección de Datos 30

3.3. Análisis e interpretación de la información 31

CAPÍTULO IV: IMPLEMENTACIÓN COMPUTACIONAL 32

4.1. Preparación del Entorno de Desarrollo 33

4.2. Extracción y Preprocesamiento de los Datos 35

4.2.1. Descarga de documentos 35

4.2.2. Extracción de información de los documentos. 35

4.2.3. Generación de Documentos 36

4.2.4. División en fragmentos 38
4.3. Generación de Embeddings 39
4.4. Creación de la base de datos Vectorial 40
4.5. Implementación del recuperador 42
4.6. Generación de respuestas 44
4.7. Interfaz de Usuario 47
4.8. Despliegue del sistema en HugginFace 49
4.9. Evaluación 53
CAPÍTULO V: RESULTADOS 56
CAPÍTULO VI: CONCLUSIONES y RECOMENDACIONES 59
ANEXOS 63
Anexo 1: Codigo fuente app.py 63
Anexo 2: Código fuente requeriments.txt 67
BIBLIOGRAFÍA 68

INDICE DE FIGURAS

Figura 1 Conjunto Anidado de Tecnologías de Inteligencia Artificial 18
Figura 2 *Flujo simple de Recuperación Aumentada por Generación.* 22
Figura 3 *Interfaz y funcionamiento de Legal's Chatbot* 23
Figura 4 *Portal web de la Plataforma Vincúlate* 24
Figura 5 *Logo de OpenAI* 26
Figura 6 *Logo del framework Langchain* 27
Figura 7 Logotipo de la base de datos vectorial Chroma 27
Figura 8 *Logo del framework Gradio* 28
Figura 9 *Logo de la Plataforma Huggin Face* 28
Figura 10 *Arquitectura de desarrollo para el asistente virtual.* 32
Figura 11 *Lista de librerías a instalar* 33
Figura 12 *Importación de librerías para procesamiento de los datos* 33
Figura 13 *Importación de librerías para el uso de LLM e interfaz gráfica* 34
Figura 14 *Importación de credenciales de OpenAI* 34
Figura 15 *Descarga de los documentos con requests* 35
Figura 16 *Dataframe de documentos descargados* 36
Figura 17 *Lectura de contenido y generación de documents* 37
Figura 18 *Visualización de un documento generado* 37
Figura 19 *División de documentos en fragmentos* 38
Figura 20 *Visualización de los fragmentos generados* 39
Figura 21 *Instanciado la clase OpenAIEmbedding* 40
Figura 22 *Visualización de una representación vectorial* 40
Figura 23 *Generación de la base de datos vectorial* 41
Figura 24 *Visualización de metadatos de documentos relevantes recuperados* 42
Figura 25 *Implementación del Recuperador* 43
Figura 26 *Documentos recuperados y sus metadatos* 43
Figura 27 *Instanciando el modelo de lenguaje* 44
Figura 28 *Template de instrucciones del asistente* 45
Figura 29 *Creación de cadena Rag con Langchain* 46
Figura 30 *Generación de respuestas a consultas con RAG* 46
Figura 31 *Bloques en la interfaz de Gradio* 47
Figura 32 *Visualización de la interfaz en notebook* 48
Figura 33 *Botones de la interfaz* 49
Figura 34 *Clase VinculateBot contenida en app.py* 50
Figura 35 *Repositorio de Código Fuente en Huggin Face* 52
Figura 36 *Ventana del Login del asistente desplegado en Spaces* 52
Figura 37 *Ventana principal del asistente virtual desplegado en Spaces* 53
Figura 38 *Encuesta compartida con usuarios de la plataforma.* 54

INDICE DE TABLAS

Tabla 1 Resumen Promedios de Puntajes por Pregunta 56
Tabla 2 Resumen Puntaje Promedio por dimensión 57

RESUMEN

Implementación de un Asistente Virtual Basado en Inteligencia Artificial Generativa para la Plataforma Vincúlate

Renzo Osmar Salazar Carazas

SEPTIEMBRE - 2024

Asesor	:	Pérez Núñez, Jhelly Reynaluz
Grado Obtenido	:	Licenciado en Computación Científica
Mención	:	Computación Científica

Este trabajo presenta el desarrollo de un asistente virtual para la plataforma Vincúlate del Concytec, diseñado para proporcionar información sobre las herramientas que facilitan la vinculación entre la academia y la industria. Para ello, se recopilaron los documentos, guías, manuales y contenido de la plataforma. Posteriormente, los datos fueron divididos en fragmentos y almacenados en la base de datos vectorial Chroma. Mediante la Recuperación Aumentada por Generación (RAG) y el uso del modelo de lenguaje de gran tamaño GPT-3.5 turbo, junto con el framework Langchain, se generaron respuestas orientadas a este dominio específico. Las evaluaciones realizadas demostraron que el asistente generó respuestas precisas y relevantes a las preguntas de los usuarios, mejorando la accesibilidad a la información y manteniendo la coherencia del contenido. Estos resultados son alentadores y pueden ser replicados en otras bases de conocimientos del Concytec.

Palabras claves: Asistente Virtual, Inteligencia Artificial Generativa, Generación Aumentada por Recuperación, Modelos de Lenguaje de Gran Tamaño.

ABSTRACT

Implementation of a Virtual Assistant Based on Generative Artificial Intelligence for the Vinculate Platform

Renzo Osmar Salazar Carazas

SEPTEMBER - 2024

Advisor	:	Pérez Núñez, Jhelly Reynaluz
Obtained Degree	:	Licenciado en Computación Científica
Emphasis	:	Computación Científica

This study presents the development of a virtual assistant for Concytec's Vincúlate platform, designed to provide information on the tools that facilitate the link between academia and industry. For this purpose, the documents, guides, manuals and content of the platform were compiled. Subsequently, the data were divided into fragments and stored in the Chroma vector database. By means of Generation Augmented Retrieval (GAR) and the use of the large language model GPT-3.5 turbo, together with the Langchain framework, answers oriented to this specific domain were generated. Evaluations conducted demonstrated that the assistant generated accurate and relevant responses to user queries, improving information accessibility and maintaining content coherence. These results are encouraging and can be replicated in other Concytec knowledge bases.

Keywords: Virtual Assistant, Generative AI, Retrieval Augmented Generation, research, Large Language Model.

CAPÍTULO I: INTRODUCCIÓN

En la actualidad digital, la eficiencia en el acceso a la información y la gestión del conocimiento se han vuelto elementos clave para el desarrollo y la innovación en todos los sectores. En el ámbito de la ciencia, tecnología e innovación, la habilidad para vincular el conocimiento académico con las necesidades industriales es esencial para fomentar el progreso económico y social de un país

En este contexto, el Consejo Nacional de Ciencia, Tecnología e Innovación (CONCYTEC) del Perú ha desarrollado la plataforma Vincúlate, con el objetivo de fortalecer los vínculos entre la academia y la industria, facilitando la transferencia de tecnologías y conocimientos. Esta herramienta proporciona recursos valiosos para la gestión tecnológica, incluyendo una amplia gama de documentos, guías, manuales y herramientas.

Sin embargo, a medida que la cantidad de información disponible en Vincúlate crece, surge un desafío significativo: cómo hacer que esta información sea accesible y útil para los usuarios de manera eficiente. Los usuarios de la plataforma, que incluyen investigadores, empresarios y gestores de innovación, a menudo se enfrentan a la tarea de navegar a través de grandes volúmenes de recursos para encontrar la información específica que necesitan. Esta complejidad puede resultar abrumadora, especialmente para aquellos que no están familiarizados con la terminología específica o los procesos de transferencia tecnológica, lo que podría reducir la influencia de la plataforma en el ecosistema de ciencia, tecnología e innovación del país. Los sistemas tradicionales de pregunta-respuesta a menudo no están equipados para manejar esta complejidad, lo que puede llevar a respuestas inexactas o irrelevantes. Por lo tanto, es necesario desarrollar soluciones tecnológicas aplicadas que aborden este desafío y evaluar su impacto en la mejora del acceso a la información.

En respuesta a este desafío, la inteligencia artificial (IA) emerge como una solución prometedora. La IA ha revolucionado diversos campos, ofreciendo soluciones innovadoras a problemas complejos. Modelos de IA generativa, como GPT (Generative Pre-trained Transformer) y sus variantes, han demostrado una capacidad impresionante para procesar y sintetizar grandes cantidades de información, responder preguntas

complejas y adaptarse a diversos contextos y tareas. Una de las áreas donde esta tecnología ha demostrado ser especialmente efectiva es en el desarrollo de asistentes virtuales avanzados. Estos asistentes tienen el potencial de interactuar con los usuarios de manera más natural y eficiente, facilitando la navegación y el acceso a la información en plataformas como Vincúlate.

En este contexto, la presente investigación se enfoca en el desarrollo de un asistente virtual para la plataforma Vincúlate, utilizando técnicas como la generación aumentada por recuperación (RAG) para mejorar la precisión y relevancia de las respuestas generadas por los grandes modelos de lenguaje (LLM) en dominios específicos. Este sistema busca integrar técnicas avanzadas de recuperación de información con la capacidad de generación de los modelos de lenguaje, proporcionando así respuestas más precisas y contextuales a las consultas de los usuarios.

Este proyecto de investigación se organiza en seis capítulos. El primer capítulo introduce el planteamiento del problema y los objetivos de la investigación. El segundo capítulo revisa la literatura relacionada con la inteligencia artificial generativa y la recuperación aumentada por generación. El tercer capítulo describe la metodología de investigación utilizada. El cuarto capítulo presenta la implementación computacional del asistente virtual, detallando desde la preparación del entorno de desarrollo hasta el despliegue del sistema. El quinto capítulo muestra los resultados y análisis obtenidos al implementar el asistente virtual. Finalmente, el sexto capítulo discute las conclusiones y recomendaciones para futuros trabajos.

1.1. Situación Problemática

La plataforma Vincúlate de Concytec tiene como propósito facilitar la transferencia de conocimientos y tecnologías desde el ámbito de la investigación hacia el sector industrial, proporcionando herramientas y recursos orientados a fortalecer las competencias de los miembros del Sistema Nacional de Ciencia, Tecnología e Innovación (SINACTI) en la gestión tecnológica (Vincúlate, 2024).

La plataforma brinda una gran cantidad de documentos, guías, manuales, formatos de convenios y contratos que pueden resultar abrumadores para los usuarios, especialmente aquellos no familiarizados con la terminología o los procesos. Aunque la

plataforma busca ser intuitiva, herramientas como la calculadora automatizada para clasificar el nivel de madurez tecnológica y la guía de decisión para elegir el documento adecuado son útiles.

Sin embargo, encontrar información específica sobre herramientas como los niveles de madurez tecnológica (TRL), guías de paquete tecnológico y valoración, y guías de licenciamiento y negociación se convierte en un desafío debido a la complejidad y cantidad de documentos disponibles. Los usuarios enfrentan varias dificultades para acceder y utilizar eficazmente estos recursos, lo que limita la eficacia de Vincúlate en su misión de promover la transferencia tecnológica.

Los usuarios podrían beneficiarse de un sistema que proporcione respuestas más personalizadas y ajustadas a sus necesidades específicas. La implementación de un asistente virtual que utilice Inteligencia Artificial Generativa y Recuperación Aumentada por Generación (RAG) podría abordar estos desafíos al ofrecer respuestas precisas y relevantes a las consultas de los usuarios, mejorando tanto la experiencia del usuario como la efectividad de las herramientas disponibles en la plataforma Vincúlate.

1.2. Formulación del Problema

1.2.1. Problema General

PG: ¿De qué manera la implementación de un asistente virtual basado en Inteligencia Artificial Generativa mejorará el acceso y la utilización de la información sobre las herramientas de vinculación academia-industria en la plataforma Vincúlate del Concytec?

1.2.2. Problema Específicos

PE1: ¿Cómo se puede estructurar y almacenar eficientemente el contenido de la plataforma Vincúlate para su uso en un sistema de asistencia virtual?

PE2: ¿De qué forma se puede desarrollar un asistente virtual capaz de brindar respuestas precisas y adaptadas al contexto de las consultas sobre las herramientas de plataforma Vincúlate?

PE3: ¿En qué medida el asistente virtual desarrollado mejorará la relevancia y precisión de las respuestas a las consultas de los usuarios de la plataforma Vincúlate?

1.3. Objetivos de la investigación

1.3.1. Objetivos General

Desarrollar un asistente virtual para la plataforma Vincúlate de Concytec con el propósito de facilitar el acceso a la información, brindar respuestas precisas y relevantes a las consultas de los usuarios empleando técnicas de Inteligencia Artificial Generativa y Recuperación Aumentada por Generación

1.3.2. Objetivos Específicos

OE1: Crear una base de datos vectorial a partir de los contenidos y documentos de las herramientas de la plataforma Vincúlate.

OE2: Desarrollar con el framework Langchain y grandes modelos de lenguaje (LLM) un asistente virtual que permita brindar respuestas precisas y relevantes a las preguntas sobre las herramientas de Vincúlate.

OE3: Evaluar la relevancia del asistente virtual en la generación de respuestas a las consultas de los usuarios de la plataforma.

1.4. Justificación

La justificación de este trabajo se basa en la necesidad de desarrollar un asistente virtual para la plataforma Vincúlate de Concytec. Esta solución es esencial para superar las barreras de accesibilidad y hacer que las herramientas de la plataforma se utilicen de manera efectiva. En la actualidad, la gran cantidad de información dificulta que los usuarios encuentren información rápida y precisa. Al proporcionar respuestas contextuales y personalizadas, el asistente virtual no solo mejorará la experiencia del usuario, sino que también contribuye a la innovación y desarrollo tecnológico en Perú, alineándose con los objetivos del SINACTI de fortalecer las capacidades en gestión tecnológica, y servirá como un modelo replicable para otras plataformas de conocimiento del Concytec, potenciando su impacto en el ecosistema de ciencia y tecnología del país.

CAPÍTULO II: MARCO TEÓRICO

2.1. Antecedentes

En el desarrollo de los chatbots o asistentes virtuales, los primeros sistemas como ELIZA, creado por Joseph Weizenbaum en 1966, utilizaban patrones predefinidos para simular la conversación humana (Weizenbaum, 1966). Este enfoque temprano demostró que las máquinas podían imitar las interacciones humanas básicas, pese a que con obvias limitaciones en la comprensión del contexto y la flexibilidad en la respuesta. En la década de 1990, ALICE marcó un gran avance mediante la implementación de técnicas de coincidencia de patrones más avanzadas y estructuras de datos sofisticadas (Wallace, 2009). Estas mejoras permitieron a los chatbots gestionar interacciones más complejas y diversas, lo que aumentó su uso en aplicaciones prácticas como el servicio al cliente.

La creación de Watson Assistant por IBM en 2006 marcó un cambio significativo en el campo de los chatbots. Este sistema comenzó como una supercomputadora con la capacidad de responder preguntas en lenguaje natural, una habilidad que normalmente se cree es exclusiva de los humanos. En 2011, Watson se hizo famoso al vencer a dos campeones humanos en el programa de concursos "Jeopardy". Para lograr esto, Watson empleó una arquitectura conocida como DeepQA, que analizaba numerosas preguntas, generaba posibles respuestas, recopilaba pruebas y las evaluaba con cientos de algoritmos (Ferrucci et al., 2010). IBM tenía la intención de revolucionar la tecnología al desarrollar una inteligencia artificial que pudiera procesar y analizar datos no estructurados utilizando técnicas avanzadas de procesamiento del lenguaje natural (NLP) y aprendizaje automático.

La introducción de los grandes modelos de lenguaje basados en transformadores, como los descritos en (Al-Amin et al., 2024) fue el siguiente gran salto. Estos modelos, como son GPT-2 o GPT-3 desarrollados por OpenAI, han revolucionado la capacidad de los chatbots para comprender y generar lenguaje humano de manera más coherente y natural. Estos sistemas pueden producir respuestas detalladas y contextualmente relevantes utilizando grandes volúmenes de datos y técnicas de aprendizaje profundo avanzadas, mejorando significativamente la interacción humano-máquina y el potencial de los asistentes virtuales en diversas aplicaciones industriales y comerciales.

En los últimos años, la introducción de la Recuperación Aumentada por Generación (RAG) ha permitido una mejora significativa en los sistemas de preguntas y respuestas de dominios específicos, este enfoque combina grandes modelos de lenguaje y bases de datos específicas del dominio con el fin de proporcionar respuestas precisas y contextualizadas (Sharma et al., 2024). Este método es un paso importante hacia la creación de asistentes virtuales capaces de manejar de manera efectiva información compleja y específica.

Un estudio reciente de (Balakrishnan & Purwar, 2024), comparó diversos modelos de lenguaje abiertos y su integración con el método RAG utilizando datos específicos de la empresa donde subraya la importancia de evaluar el rendimiento, accesibilidad e integración de los grandes modelos de lenguaje en contextos organizacionales específicos. Las evaluaciones de la eficacia de los modelos han demostrado que la precisión y eficiencia de los sistemas RAG pueden mejorarse significativamente cuando se combinan con técnicas de incrustación efectivas.

Se ha demostrado en trabajos previos como el "Desarrollo de chatbot de consulta dental utilizando Retrieval Augmented LLM" que la implementación de RAG puede mejorar significativamente la accesibilidad y precisión de las consultas dentales, facilitando un mejor servicio al usuario Park, J. (2024). Asimismo, según la evaluación de la herramienta Tropipay, una plataforma de pago, el proyecto "I-AVATAR: Asistente Virtual Basado en IA para un Soporte Efectivo" (Sánchez Rivero et al., 2024) ha demostrado resultados prometedores en el soporte técnico y la atención al cliente.

Estos antecedentes brindan un contexto sólido para el desarrollo de un asistente virtual para la plataforma Vincúlate, destacando la relevancia y aplicabilidad de la metodología RAG para mejorar el acceso a la información y la experiencia del usuario.

2.2. Bases Teóricas

2.2.1. Inteligencia Artificial Generativa

La inteligencia artificial generativa (IA generativa) es una rama de la inteligencia artificial que se enfoca en la capacidad de las máquinas para crear contenido nuevo y original a partir de datos preexistentes. Este tipo de inteligencia artificial emplea modelos

avanzados, como los grandes modelos de lenguaje (LLMs), para generar texto, imágenes, música y otros tipos de medios. Los recientes avances en IA generativa han transformado múltiples campos, desde la creación de contenido digital hasta el desarrollo de sistemas de asistencia virtual (AWS, 2023; NVIDIA, 2023a)

Principios Fundamentales de la IA Generativa

La IA generativa se basa en modelos de lenguaje que han sido entrenados con grandes cantidades de datos textuales. Estos modelos, como GPT-3 y GPT-4 de OpenAI, utilizan técnicas de aprendizaje profundo para comprender y generar lenguaje natural. La estructura fundamental de estos modelos es el Transformer, el cual facilita el procesamiento y la generación de texto de forma eficiente y coherente.

Los modelos de IA generativa funcionan aprendiendo patrones y estructuras en los datos de entrenamiento, lo que les permite generar contenido que es coherente y relevante en varios contextos. Por ejemplo, un modelo de lenguaje puede completar una oración, escribir un artículo o incluso mantener una conversación fluida con un usuario (NVIDIA, 2023b; AWS, 2023).

Tiene aplicaciones en una amplia variedad de campos:

- Creación de Contenido: Generación de artículos, historias, poesía y contenido multimedia.
- Asistentes Virtuales: Creación de chatbots y asistentes virtuales que son capaces de interactuar de forma natural con los usuarios y proporcionar respuestas precisas.
- Diseño y Creatividad: Creación de imágenes, diseños gráficos y música original.
- Educación y Capacitación: Generación de materiales educativos y simulaciones interactivas para la formación.

Ventajas y desafíos

La principal ventaja de la IA generativa es su capacidad para producir grandes volúmenes de contenido de alta calidad en poco tiempo. Esto es especialmente útil en la

creación de contenido digital y en la automatización de tareas que requieren generación de texto o multimedia (Brynjolfsson et al., 2023).

Sin embargo, la IA generativa también enfrenta desafíos significativos:

- Los modelos pueden quedar desactualizados si no se entrenan continuamente con nuevos datos.
- Los modelos pueden generar información incorrecta o engañosa, un fenómeno conocido como "alucinaciones".
- La dificultad para acceder a información específica de un dominio
- Entrenar modelos de IA generativa requiere enormes cantidades de datos y recursos computacionales.

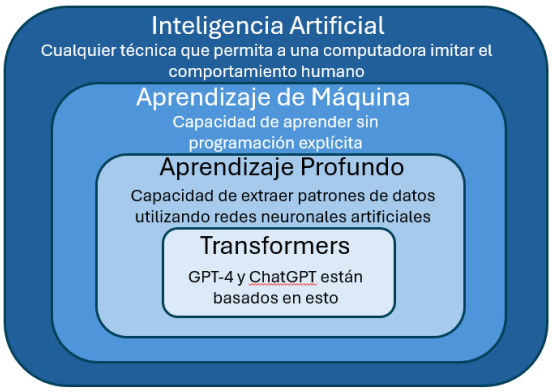
2.2.2. Grandes Modelos de Lenguaje

Los grandes modelos de lenguaje (LLMs, por sus siglas en inglés) representan un tipo de inteligencia artificial diseñada para procesar y generar lenguaje natural con alta precisión. Estos modelos se entrenan utilizando vastas cantidades de datos textuales y utilizan arquitecturas de redes neuronales avanzadas, como el Transformer, para comprender y generar texto de manera coherente y en contextos específicos (NVIDIA, 2023c; Brown et al., 2020).

Según Caelen y Blete (2023), Los LLMs, como GPT-3 y GPT-4 desarrollados por OpenAI, funcionan mediante el aprendizaje de patrones y estructuras en los datos textuales de entrenamiento. Estos modelos contienen miles de millones de parámetros ajustables que permiten captar matices complejos del lenguaje humano. Utilizan técnicas de aprendizaje profundo para procesar el texto de entrada y generar respuestas que son contextualmente relevantes y gramaticalmente correctas.

Figura 1

Conjunto Anidado de Tecnologías de Inteligencia Artificial



Arquitectura Transformer

La arquitectura Transformer es el núcleo de muchos LLMs. Introducida por Vaswani et al. en 2017, esta arquitectura utiliza mecanismos de atención para gestionar la relación entre diferentes partes del texto, a diferencia de los modelos anteriores que procesaban el texto secuencialmente, los transformadores procesan todas las palabras de una oración simultáneamente. Esto permite a los LLMs capturar relaciones a largo plazo entre palabras y comprender el contexto de manera más efectiva, permitiendo al modelo enfocarse en las palabras más relevantes para la tarea específica. Esta capacidad de atención es lo que permite a los LLMs manejar dependencias a largo plazo en el texto y generar contenido coherente.

Aplicaciones y Limitaciones

Los LLMs tienen una amplia gama de aplicaciones en diversas industrias:

- Asistentes virtuales y chatbots: Los LLMs son la tecnología subyacente de muchos asistentes virtuales y chatbots modernos, lo que les permite mantener conversaciones más naturales, comprender mejor las consultas de los usuarios y proporcionar respuestas más precisas y relevantes.
- Generación de contenido: Los LLMs pueden generar diferentes tipos de contenido creativo, como artículos, historias, poemas, guiones e incluso código de programación, lo que los convierte en herramientas valiosas para escritores, redactores y desarrolladores.

- Traducción de idiomas: La capacidad de los LLMs para comprender y generar texto en diferentes idiomas los hace ideales para la traducción automática, mejorando la precisión y la fluidez en comparación con los métodos tradicionales.
- Análisis de sentimientos: Los LLMs pueden analizar grandes volúmenes de texto para determinar el sentimiento general expresado, lo que los hace útiles para aplicaciones como la monitorización de redes sociales, la investigación de mercados y la atención al cliente.

En Minae et al., 2024, se revisan diferentes arquitecturas de LLMs y su evolución desde GPT-1 hasta GPT-4, destacando los avances y los retos que han enfrentado estos modelos a lo largo del tiempo. A pesar de sus capacidades avanzadas, los LLMs presentan varias limitaciones. Una de las principales preocupaciones es el sesgo inherente en los datos de entrenamiento, que puede llevar a la generación de respuestas discriminatorias o parciales. Además, estos modelos pueden generar información inexacta o inventar datos, lo que puede ser problemático en aplicaciones críticas. Los altos costos computacionales asociados con el entrenamiento y despliegue de LLMs también son una barrera significativa para su adopción generalizada.

2.2.3. Recuperación Aumentada por Generación (RAG)

La Recuperación Aumentada por Generación (RAG) es una técnica avanzada en el ámbito del procesamiento de lenguaje natural que integra la recuperación de información con la generación de texto. Esta innovadora aproximación integra grandes modelos de lenguaje (LLMs) como GPT-3.5-turbo o GPT-4 con bases de datos o conjuntos de documentos externos, permitiendo generar respuestas más precisas y contextuales. RAG utiliza conceptos de vectorización, que es el proceso de convertir datos textuales en representaciones numéricas (vectores) que capturan la semántica del contenido, y búsqueda semántica, un método que permite encontrar información relevante basándose en el significado de los datos, para retroalimentar los LLMs con información que no se incluyó durante su fase de entrenamiento, mejorando así la calidad de las respuestas y reduciendo comportamientos no deseados como las alucinaciones , es decir, la generación de información falsa o engañosa que el modelo presenta como verdadera

debido a patrones erróneos aprendidos durante su entrenamiento. Esta combinación de recuperación y generación no solo aumenta la precisión, sino que también mejora significativamente la experiencia del usuario al proporcionar respuestas más relevantes y fundamentadas (Lewis et al., 2021; AWS, 2023; NVIDIA, 2023d).

Principios Fundamentales de la RAG

La RAG se apoya en varios componentes clave:

- Base de Conocimiento: Este elemento contiene el conocimiento objetivo del cual el LLM no es consciente y que tiene un alto riesgo de alucinación. La información puede estar en diversos formatos como PDF, Excel, Word, y necesita ser indexada para su uso eficaz. Esta base de conocimiento actúa como la fuente de datos estática que se consulta para recuperar información relevante y precisa.
- Modelo de Embeddings: Un embedding es una representación vectorial generada por una red neuronal entrenada sobre un conjunto de datos. Este modelo convierte los fragmentos de texto en vectores numéricos que capturan la semántica del contenido, permitiendo así la comparación y recuperación eficiente de la información relevante. Los embeddings son esenciales para indexar y buscar en la base de conocimiento.
- Base de Datos Vectorial: Aquí se guarda e indexa la información vectorial de los fragmentos mediante los embeddings. Esta base de datos es crucial para la recuperación rápida y precisa de la información relevante. Las bases de datos vectoriales utilizan técnicas de indexación que permiten realizar búsquedas semánticas eficientes.
- LLM: El modelo de lenguaje generativo que emplea la información recuperada para producir respuestas precisas y adecuadas al contexto. Los LLMs, como GPT-3 y GPT-4, procesan los fragmentos obtenidos y generan respuestas que no solo integran la información, sino que también la contextualizan de manera efectiva, ofreciendo así una experiencia de usuario más coherente y útil.

Arquitectura RAG

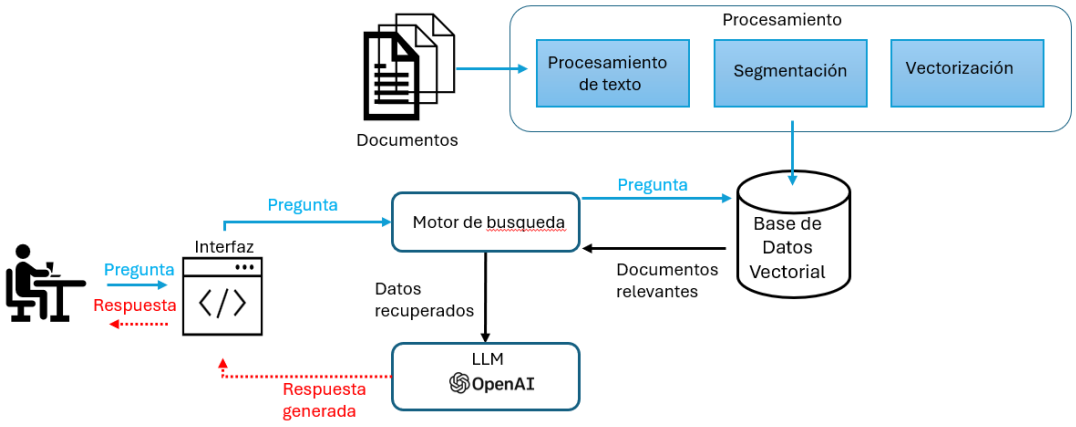
Un sistema RAG típico consta de tres componentes principales: una base de conocimiento, un modelo de recuperación de información y un modelo de lenguaje generativo como se puede observar en la Figura 2. La base de conocimiento alberga la información específica del dominio, que puede incluir documentos, artículos, bases de datos, etc. El modelo de recuperación de información selecciona la información más relevante de la base de conocimiento en función de la consulta del usuario, generalmente utilizando técnicas de similitud semántica y embeddings. El modelo de lenguaje generativo, como un LLM, utiliza la información recuperada como contexto para generar una respuesta precisa y coherente a la consulta del usuario.

El funcionamiento a alto nivel de un sistema RAG es el siguiente:

- El usuario realiza una pregunta o solicitud al sistema.
- La pregunta se transforma en un vector utilizando el mismo sistema de embedding que se utilizó para indexar la base de datos.
- Se calculan las distancias entre la pregunta y todos los vectores en la base de datos.
- Se seleccionan los fragmentos más relevantes, los fragmentos de información recuperados se integran en la entrada del modelo de lenguaje generativo, junto con la pregunta original.
- El LLM utiliza el prompt, que incluye la pregunta y el contexto recuperado, para generar una respuesta basada en la información más relevante.

Figura 2

Flujo simple de Recuperación Aumentada por Generación.



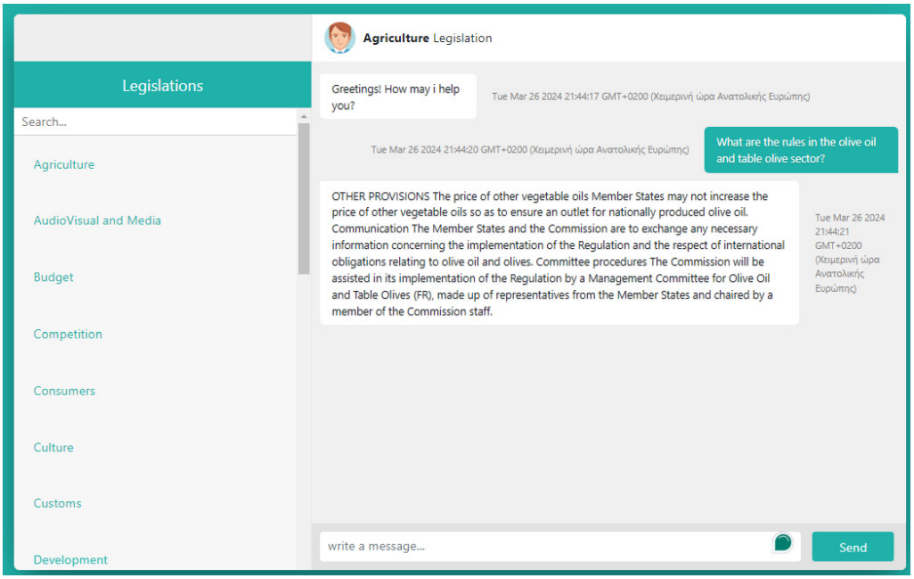
Este enfoque permite a los LLMs proporcionar respuestas más precisas y reducir el riesgo de alucinaciones, mejorando así la utilidad y fiabilidad de los sistemas de IA generativa además evita la necesidad de reentrenar los LLMs, lo que puede ser costoso.

Aplicaciones en sistemas de QA y chatbots

La técnica RAG ha demostrado ser efectiva en la implementación de sistemas de preguntas y respuestas (QA) y en el desarrollo de chatbots, fortaleciendo su habilidad para gestionar preguntas complejas y detalladas. Por ejemplo, Kapsalis (2024) desarrolló un chatbot para la legislación de la UE utilizando RAG, lo que permitió proporcionar respuestas precisas basadas en documentos legislativos. Borek (2024) evaluó comparativamente enfoques basados en LLM para la creación de chatbots, destacando la efectividad de RAG en entornos empresariales. Bink (2023) exploró cómo las respuestas personalizadas con IA generativa, mejoradas con RAG, pueden mejorar la interacción con el cliente, utilizando aprendizaje de cero disparos para adaptarse a diversas consultas sin necesidad de entrenamiento adicional.

Figura 3

Interfaz y funcionamiento de Legal's Chatbot



Nota. La interfaz del chatbot permitirá a los usuarios buscar legislación específica, recibir explicaciones de ideas legales y obtener asesoramiento sobre las necesidades de cumplimiento. Fuente. Kapsalis, A. (2024). *Legal chatbot's results* [Captura de pantalla de tesis de maestría]. En *Chatbot for EU legislation* (p. [52]). Repositorio Institucional IHU. https://repository.ihu.edu.gr//xmlui/handle/11544/30419

2.2.4. Plataforma Vincúlate

La plataforma Vincúlate fue desarrollada por el Consejo Nacional de Ciencia, Tecnología e Innovación (CONCYTEC) del Perú. Está diseñada para facilitar la vinculación entre la academia y la industria, promoviendo la transferencia de tecnología y conocimientos. La plataforma ofrece diversas herramientas y recursos, incluyendo guías, manuales y formatos de convenios, destinados a fortalecer las capacidades de los miembros del Sistema Nacional de Ciencia, Tecnología e Innovación (SINACTI) en gestión tecnológica. Vincúlate busca simplificar los procesos de colaboración y transferencia de conocimientos, contribuyendo así al desarrollo tecnológico e innovación en el Perú (Vincúlate, 2024).

Figura 4

Portal web de la Plataforma Vincúlate



Nota. El gráfico muestra las herramientas que ofrece la Plataforma Vincúlate y en particular la sección de Contratos para servicios de I+D+I. Fuente. Vincúlate. (2024) *Portal web de la Plataforma Vincúlate* [Captura de pantalla]. CONCYTEC. https://vinculate.concytec.gob.pe/convenios-y-contratos/contratos-para-servicios-id/

VINCÚLATE ofrece las siguientes herramientas a gestores, investigadores, innovadores, tecnólogos, autoridades, emprendedores, estudiantes y la población en general:

Niveles de madurez tecnológica (Techonology Readiness Level o TRL): Esta herramienta es ampliamente utilizada para evaluar el grado de desarrollo de las investigaciones tecnológicas. Esta herramienta ha sido adaptada a partir de los niveles propuestos inicialmente por la NASA y el programa Horizonte 2020 de la Unión Europea, con el propósito de proporcionar un marco estandarizado que permita determinar el estado de madurez de tecnologías emergentes. En este contexto, se ha desarrollado un servicio en línea basado en una calculadora automatizada que facilita a los equipos técnicos la clasificación de sus proyectos e investigaciones dentro de los diferentes niveles de madurez tecnológica. Este servicio no solo permite una evaluación precisa del estado de desarrollo de las tecnologías, sino que también sirve como un apoyo en la planificación estratégica para avanzar hacia etapas de mayor desarrollo y comercialización.

Guía de paquete tecnológico y valoración: Es un documento técnico fundamental que ofrece formatos y herramientas específicas para la creación de un paquete tecnológico robusto. Este documento proporciona métodos detallados para la valorización de tecnologías, junto con análisis económicos exhaustivos y evaluaciones del mercado potencial. Además, aborda conceptos clave en la transferencia o comercialización de tecnologías, proporcionando un marco integral que ayuda a investigadores y desarrolladores a maximizar el valor de sus innovaciones. La guía también incluye estrategias de evaluación que consideran tanto los aspectos económicos como los factores de mercado, con el fin de asegurar que las tecnologías desarrolladas sean competitivas y respondan adecuadamente a las necesidades del mercado.

Guía de licenciamiento y negociación: Este documento técnico ofrece formatos, convenios, contratos y la información necesaria para llevar a cabo una negociación exitosa, con el objetivo de lograr el licenciamiento de tecnologías y fomentar la colaboración entre las distintas partes involucradas.

Guía Metodológica Tangüis: Este documento técnico está diseñado con el objetivo de facilitar la comprensión y la formación de alianzas estratégicas entre potenciales socios, incluyendo universidades, centros de investigación y entidades públicas del sector empresarial. La guía ofrece formatos editables para convenios, contratos, cláusulas y acuerdos, proporcionando un marco práctico y adaptable para la creación de alianzas efectivas. Además, se ha desarrollado una guía de decisión automatizada que ayuda a los usuarios a seleccionar el documento más adecuado según sus necesidades específicas, optimizando así el proceso de negociación y colaboración. Esta guía metodológica es un recurso valioso para las entidades que buscan fortalecer sus capacidades de cooperación y colaboración a través de acuerdos estratégicos bien estructurados y alineados con sus objetivos de investigación y desarrollo.

A pesar de que la plataforma Vincúlate proporciona una amplia gama de herramientas y recursos, la abundancia y diversidad de documentos en distintos formatos como Word y PDF pueden dificultar el acceso y uso eficiente de la información para los usuarios. Este escenario evidencia la necesidad de soluciones que simplifiquen la interacción con la plataforma. En este contexto, el desarrollo de un asistente virtual se

presenta como una oportunidad para mejorar la accesibilidad y eficiencia en la gestión de la información, facilitando así el aprovechamiento pleno de los recursos que Vincúlate ofrece.

2.2.5. Tecnologías de Desarrollo

A continuación, se detallan las tecnologías usadas en el desarrollo del asistente virtual.

OpenAI API

La API de OpenAI, cuyo logo se presenta en la Figura 5, proporciona acceso a modelos de lenguaje de gran escala como GPT-3.5 y GPT-4, permitiendo a los desarrolladores integrar capacidades avanzadas de procesamiento del lenguaje natural en sus aplicaciones. Esta API ofrece una variedad de endpoints para tareas como completar texto, responder preguntas y generar contenido creativo. La flexibilidad de la API permite controlar factores como la temperatura y la cantidad máxima de tokens para controlar la creatividad y la longitud de las respuestas generadas (OpenAI, 2024).

Figura 5

Logo de OpenAI



Fuente. OpenAI, (2024)

Framework Langchain

La Figura 6 presenta el logotipo de Langchain, un framework de código abierto diseñado para facilitar el desarrollo de aplicaciones basadas en modelos de lenguaje de gran escala (LLMs). Con su amplia gama de componentes modulares y abstracciones, los desarrolladores pueden crear aplicaciones complejas de manera más eficiente. Las herramientas de Langchain permiten la gestión de la memoria en las conversaciones, la integración con fuentes de datos externas y el manejo de prompts. Una de sus características más destacadas es su capacidad para implementar fácilmente el método de Recuperación Aumentada por Generación (RAG), mejorando la precisión y contextualización de las respuestas generadas por los LLMs (Langchain, 2024).

Figura 6

Logo del framework Langchain



Fuente. LangChain, (2024)

Chroma

Chroma, podemos observar su logo en la Figura 7, es una base de datos vectorial de código abierto diseñada específicamente para aplicaciones de inteligencia artificial. La capacidad de almacenar y recuperar vectores de alta dimensionalidad lo convierte en una herramienta ideal para trabajar con embeddings de texto y otros tipos de datos. El rendimiento, la facilidad de uso y la capacidad de escalar de Chroma lo convierten en una opción popular para implementar sistemas de recuperación de información en aplicaciones de IA generativa (Chroma, 2024).

Figura 7

Logotipo de la base de datos vectorial Chroma



Fuente. Chroma, (2024)

Gradio

Gradio es una biblioteca de código abierto que facilita la creación rápida de interfaces web para modelos de aprendizaje automático y aplicaciones de inteligencia artificial. Permite a los desarrolladores crear demostraciones interactivas y prototipos de manera sencilla, sin necesidad de conocimientos profundos en desarrollo web. Gradio, cuyo logo se visualiza en la Figura 8, es particularmente útil para visualizar y compartir modelos de procesamiento de lenguaje natural, visión por computadora y otros tipos de inteligencia artificial, lo que permite a los usuarios interactuar directamente con estos modelos a través de una interfaz fácil de entender (Gradio, 2024).

Figura 8

Logo del framework Gradio



Fuente. Gradio, (2024)

Huggin Face Spaces

Hugging Face Spaces, representado con su logo en la Figura 9, es una plataforma que permite a desarrolladores e investigadores crear, alojar y compartir aplicaciones de inteligencia artificial de manera sencilla y gratuita. Facilita la demostración de modelos de aprendizaje automático y aplicaciones de inteligencia artificial sin la necesidad de instalar y configurar una infraestructura compleja. Spaces soporta frameworks como Gradio para crear interfaces interactivas, y se integra directamente con el ecosistema de Hugging Face, permitiendo el uso de modelos pre-entrenados y datasets disponibles en la plataforma. (Hugging Face, 2024).

Figura 9

Logo de la Plataforma Huggin Face



Fuente. Hugging Face, (2024)

CAPÍTULO III: METODOLOGÍA

3.1. Tipo y Diseño de Investigación

Este trabajo presenta el desarrollo de un asistente virtual para la plataforma Vincúlate del Concytec, cuyo objetivo es facilitar el acceso a la información y mejorar la experiencia de los usuarios en temas de vinculación entre academia e industria. El diseño de la investigación es aplicado y descriptivo, ya que se centra en la creación de una solución tecnológica (el asistente virtual) y en la evaluación de su rendimiento a través de interacciones directas con los usuarios de la plataforma.

Se pretende evaluar la precisión, relevancia, coherencia y nivel de satisfacción de las respuestas generadas por el asistente virtual, basadas en la información contenida en los documentos de la plataforma Vincúlate. Los usuarios participantes, quienes son expertos en el campo de la transferencia tecnológica y usuarios frecuentes de la plataforma, interactuarán con el asistente para proporcionar una evaluación cualitativa y cuantitativa de su desempeño.

3.1.1. Unidad de Análisis

Las unidades de análisis incluyen los documentos, guías, manuales y demás contenidos disponibles en la plataforma Vincúlate, así como las consultas realizadas por los usuarios (investigadores, especialistas del área y gestores tecnológicos) de la plataforma.

3.1.2. Población de Estudio

La población de estudio abarca todos los documentos y contenidos de la plataforma Vincúlate. También se incluyen las interacciones de los usuarios con la plataforma, específicamente los investigadores, especialistas y gestores tecnológicos que utilizan Vincúlate.

3.1.3. Tamaño de Muestra

Se seleccionó todos los documentos (guías, manuales, convenios, acuerdos), asegurando que cubran todas las herramientas y tipos de contenido disponibles en la plataforma Vincúlate. Además, se encuestó a 40 participantes, compuestos por

investigadores, especialistas del área y gestores tecnológicos, todos ellos usuarios frecuentes de la plataforma Vincúlate.

3.1.4. Selección de Muestra

No se aplicará ninguna técnica de muestreo para los documentos, ya que se trabajará con el conjunto completo de los recursos disponibles en la plataforma Vincúlate. Para los usuarios, se seleccionó una muestra de 20 participantes, compuesta por investigadores, especialistas del área y gestores tecnológicos, todos ellos usuarios frecuentes de la plataforma Vincúlate. La elección de esta muestra se fundamenta en el concepto de juicio de expertos, que es una metodología válida para estudios donde se busca la opinión de individuos con alto nivel de conocimiento en el tema en cuestión. De acuerdo con Guest et al. (2006), en estudios cualitativos donde se exploran percepciones o la efectividad de una herramienta, la saturación de información puede lograrse con un número relativamente pequeño de participantes, especialmente si estos poseen experiencia relevante en el área evaluada.

3.2. Técnicas de Recolección de Datos

Para la recolección de datos se utilizó una estrategia sistemática que incluye las siguientes etapas y métodos:

1. Descarga de Recursos: Se descargaron todos los recursos disponibles en la web de la plataforma Vincúlate, incluyendo una variedad de documentos, manuales y guías relacionados con las herramientas y servicios ofrecidos por la plataforma.
2. Generación de documentos: La información de las páginas web correspondientes a las herramientas de la plataforma Vincúlate fue consolidada en documentos específicos para su posterior análisis. Esta etapa involucró la extracción de información relevante y su estructuración en documentos PDF.
3. Centralización de la información de los documentos: Toda la información recolectada se centralizó en un archivo de Excel para facilitar su organización y análisis. Este archivo contiene detalles sobre cada documento descargado, como el título del documento, tipo de documento, herramienta a la que pertenece y url de donde proviene.

4. Carga de la información: Todos los documentos descargados y generados en las etapas anteriores, fueron almacenados en un repositorio de Dropbox.
5. Técnicas de Recolección Adicionales: Se distribuyó una encuesta, desarrollada en Google forms, a los usuarios de la plataforma para recopilar datos sobre su experiencia y relevancia ofrecida por el asistente virtual.

3.3. Análisis e interpretación de la información

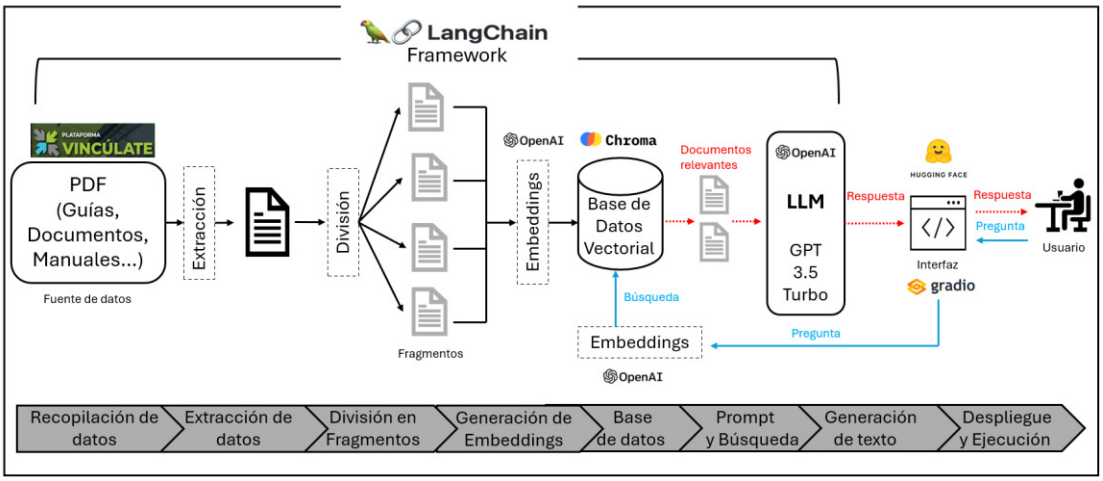
Los documentos recolectados serán procesados con herramientas de Python, como PyPDF para leer el contenido y RecursiveCharacterTextSplitter para dividir la información. Además, se utilizarán modelos de embeddings para vectorizar la información. Las encuestas se analizarán mediante técnicas de análisis de datos, incluyendo estadística descriptiva para evaluar la precisión y coherencia de las respuestas del asistente virtual. Los resultados se interpretarán en función de su capacidad para mejorar el acceso a la información y la experiencia del usuario en la plataforma Vincúlate.

CAPÍTULO IV: IMPLEMENTACIÓN COMPUTACIONAL

En este capítulo se describe la metodología empleada para la implementación computacional del asistente virtual, utilizando Recuperación Aumentada por Generación (RAG). Esta implementación se estructura en varias etapas clave: la preparación del entorno de desarrollo, que incluye la configuración de herramientas y librerías; procesamiento de los datos, la construcción de una base de datos vectorial mediante la generación de embeddings y su almacenamiento en Chroma; la implementación del sistema de recuperación y generación, que abarca el desarrollo de un modelo de recuperación de contexto, la contextualización de preguntas y la generación de respuestas relevante utilizando grandes modelos de lenguaje (LLM); el desarrollo de una interfaz de usuario intuitiva con Gradio para facilitar la interacción con el asistente; y el despliegue del sistema en Hugging Face Spaces, asegurando su disponibilidad y funcionalidad. Además, se evalúan los resultados obtenidos mediante pruebas exhaustivas, y un cuestionario realizado a los usuarios de la plataforma.

Figura 10

Arquitectura de desarrollo para el asistente virtual.



Nota. La figura muestra la arquitectura detallada del desarrollo del asistente virtual, tal como se describió en las etapas anteriores

4.1. Preparación del Entorno de Desarrollo

En esta etapa, se detallarán los pasos requeridos para establecer el entorno de desarrollo, lo que incluye la instalación de las librerías fundamentales y la configuración de las credenciales necesarias.

Para comenzar, es crucial instalar todas las librerías necesarias para el procesamiento de datos, la creación de la base de datos vectorial, y la implementación del sistema RAG y la interfaz de usuario. La instalación se realiza mediante el uso de pip, el gestor de paquetes de Python. En la Figura 11, se presenta el código utilizado para la instalación de estas librerías:

Figura 11

Lista de librerías a instalar

```
[ ] from IPython.display import clear_output

[ ] !pip install langchain langchain_openai pandas langchain-community langchain_chroma openai tiktoken chromadb pypdf gradio python-dotenv
    clear_output()
```

Luego de instalar las librerías, se procede a la importación de estas librerías necesarias para el desarrollo del asistente, estás las dividiremos en dos grupos que son los siguientes:

1. Librerías para el Procesamiento de Datos y Creación de la Base de Datos Vectorial, como se muestra en la Figura 12, estas librerías son esenciales para la manipulación de datos, la creación de embeddings y el almacenamiento en la base de datos vectorial.

Figura 12

Importación de librerías para procesamiento de los datos

```
#LIBRERÍAS PARA PROCESAMIENTO DE DATOS Y CREACIÓN DE LA BASE DE DATOS VECTORIAL
import os
import pandas as pd
import requests
from pypdf import PdfReader
from langchain.docstore.document import Document
from langchain_openai import OpenAIEmbeddings
from langchain_text_splitters import CharacterTextSplitter
from langchain_chroma import Chroma
```

2. Librerías para Implementar el Sistema RAG y la Interfaz, la Figura 13 muestra las librerías que se utilizan para implementar el modelo de recuperación y generación, así como la interfaz de usuario. Incluyen componentes para la creación de prompts, la integración con grandes modelos de lenguaje (LLM), y la creación de la interfaz con Gradio.

Figura 13

Importación de librerías para el uso de LLM e interfaz gráfica

```
#LIBRERÍAS PARA IMPLEMENTAR EL SISTEMA RAG Y LA INTERFACE
from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from langchain.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain.vectorstores import Chroma
from langchain_core.output_parsers import StrOutputParser
from langchain.schema import AIMessage, HumanMessage
from langchain_core.runnables import RunnablePassthrough, RunnableParallel
import gradio as gr
```

Además, para utilizar los modelos de OpenAI este nos proporciona un token para el uso de su API, es necesario configurar esta credencial de manera adecuada ya que se incurre en costos. La opción que hemos tomado es definirla en una variable de entorno, manera por la cual si alguien revisa el código no visualizará dicho token de permisos.

Figura 14

Importación de credenciales de OpenAI

```
!echo "OPENAI_API_KEY=sk-proj-uq01gV2aJIr[redacted]kFJePM0PjQwLIXkJKEK042Q" >> .env

# Cargar las credenciales de OpenAI
from dotenv import load_dotenv
import os

# Cargar las variables de entorno desde el archivo .env
load_dotenv()

# Configurar las credenciales de OpenAI
api_key = os.getenv("OPENAI_API_KEY")
```

En la Figura anterior se muestra cómo configurar y acceder a las credenciales de OpenAI.

4.2. Extracción y Preprocesamiento de los Datos

En esta etapa se detallará los pasos llevados a cabo que incluye la descarga de los documentos recolectados de la plataforma Vinculate, la extracción de información de estos y la generación del objeto Document con los metadatos correspondientes.

4.2.1. Descarga de documentos

El primer paso consiste en descargar los documentos que contienen la información relevante de la plataforma desde una URL específica. Para ello, se utiliza el módulo requests de Python. La siguiente Figura 15 ilustra el proceso de descarga:

Figura 15

Descarga de los documentos con requests

```
#url que contiene los documentos comprimidos de la pagina web:https://vinculate.concytec.gob.pe/

import requests
import os

# URL de descarga directa
url= "https://www.dropbox.com/scl/fi/rmnzwulo6qluj8c91hv10/database.rar?rlkey=3mwc76ia2qtbx3odfnw65bkip&st=0z3bgvcr&dl=1"

# Nombre del archivo local
filename = "database.rar"

# Descargar el archivo
response = requests.get(url)
if response.status_code == 200:
    with open(filename, 'wb') as file:
        file.write(response.content)
    print(f"Archivo {filename} descargado correctamente.")
else:
    print("Error al descargar el archivo.")

# Verificar que el archivo se ha descargado
if os.path.exists(filename):
    print(f"El archivo {filename} está en el directorio actual.")
    print(f"Tamaño del archivo: {os.path.getsize(filename)} bytes")
else:
    print(f"No se pudo encontrar el archivo {filename}")
```

```
Archivo database.rar descargado correctamente.
El archivo database.rar está en el directorio actual.
Tamaño del archivo: 24194762 bytes
```

En este paso se llegó a descargar 33 documentos en formato PDF que contienen un total de 651 páginas de información brindada por la plataforma Vincúlate.

4.2.2. Extracción de información de los documentos.

En la etapa de recolección de datos se comentó que la centralización de la información de los documentos se encontraba en un documento de Excel, el cual contenía los datos de los documentos recolectados. Se utilizó pandas para leer los datos de este

archivo, dichos datos servirán más adelanta como metadatos para la generación de los documents (objeto perteneciente a la clase Document de Langchain). En la figura 16 se previsualiza el contenido de este archivo.

Figura 16

Dataframe de documentos descargados

```
#Visualizamos una muestra del archivo con los metadatos de los documentos
import pandas as pd
metadata = pd.read_excel('/content/data/database/database_vinculate.xlsx',engine='openpyxl')
metadata.head(10)
```

	titulo_documento	Herramienta	tipo_documento	nombre_archivo	url
0	Contenido Web Acerca de Vinculate	ACERCA DE VINCULATE	Contenido Web	ACERCA_DE_VINCULATE.pdf	https://vinculate.concytec.gob.pe/acerca-de-vi...
1	Contenido Web Convenios y Contratos	CONVENIOS Y CONTRATOS	Contenido Web	CONVENIOS _Y_CONTRATOS.pdf	https://vinculate.concytec.gob.pe/convenios-y-...
2	Contenido Web Acuerdos de Confidencialidad	CONVENIOS Y CONTRATOS	Contenido Web	ACUERDOS_DE_CONFIDENCIALIDAD.pdf	https://vinculate.concytec.gob.pe/acuerdos-de-...
3	Contenido Web Contratos para servicios I+D+i	CONVENIOS Y CONTRATOS	Contenido Web	CONTRATOS_PARA_SERVICIOS_IDI.pdf	https://vinculate.concytec.gob.pe/convenios-y-...
4	Contenido Web Convenios de colaboración	CONVENIOS Y CONTRATOS	Contenido Web	CONVENIOS_DE_COLABORACION.pdf	https://vinculate.concytec.gob.pe/convenios-y-...
5	Contenido Web Acuerdos de transferencia de mat...	CONVENIOS Y CONTRATOS	Contenido Web	ACUERDO_DE_TRANSFERENCIA_DE_MATERIALES.pdf	https://vinculate.concytec.gob.pe/convenios-y-...
6	Contenido Web Cláusulas especiales	CONVENIOS Y CONTRATOS	Contenido Web	CLAUSULAS_ESPECIALES.pdf	https://vinculate.concytec.gob.pe/convenios-y-...
7	Guía metodológica Tangüis	CONVENIOS Y CONTRATOS	Guía	Guia_metodologica_TANGUIS.pdf	https://vinculate.concytec.gob.pe/download/Gui...
8	D1C - Acuerdo de confidencialidad	CONVENIOS Y CONTRATOS	Acuerdos de confidencialidad	D1C_-_Acuerdo_de_confidencialidad_y_no_divulga...	https://vinculate.concytec.gob.pe/download/D1C...

4.2.3. Generación de Documentos

El siguiente paso fue generar objetos Document[1] para cada archivo PDF, incluyendo los metadatos asociados en el dataframe cargado previamente. Se utiliza PdfReader para leer el contenido de los archivos PDF y Document de LangChain para estructurar la información. Se presenta el código correspondiente en la Figura 17:

[1] Para obtener más información sobre el proceso de generación de objetos Document, consulte https://js.langchain.com/v0.1/docs/modules/data_connection/document_loaders/creating_documents/ .

Figura 17

Lectura de contenido y generación de documents

```
[ ] #GENERAMOS NUESTROS DATOS EN FORMATO DEL OBJETO DOCUMENT RECORRIENDO TODOS LOS DOCUMENTOS EN EL FOLDER
    #además le agregamos metadatos en base a nuestro dataframe
    import os
    from pypdf import PdfReader
    from langchain.docstore.document import Document
    #ruta donde se encuentran los pdfs
    pdf_folder_path = "/content/data/database/"
    documents = []
    page_counts = {}
    for file in os.listdir(pdf_folder_path):
      #generamos la ruta donde se encuentra nuestro archivo
      source_file=pdf_folder_path+file
      print(source_file)
      if os.path.isfile(source_file) and file.endswith('.pdf'):
        reader = PdfReader(source_file)#Lector de pdf
        print(len(reader.pages))
        i=0
        page_count = len(reader.pages)
        page_counts[file] = page_count  # Guardar el conteo de páginas para este archivo
        for page in reader.pages:
            documents.append(Document(page_content=page.extract_text(),
                              metadata={'source':file,'page':i,
                                        'titulo':get_field_by_filename(file, "titulo_documento", metadata),
                                        'herramienta':get_field_by_filename(file, "Herramienta", metadata),
                                        'tipo_documento':get_field_by_filename(file, "tipo_documento", metadata),
                                        'url':get_field_by_filename(file,"url",metadata)}))
            i += 1

/content/data/database/ACERCA_DE_VINCULATE.pdf
2
/content/data/database/Casos_Paquetes_Tecnologicos_Dilutor_Coloide.pdf
4
/content/data/database/Clusula_especial_I.pdf
2
```

Se mostrará el extracto de un documento para conocer su estructura en la siguiente figura:

Figura 18

Visualización de un documento generado

```
documents[0]

Document(metadata={'source': 'ACERCA_DE_VINCULATE.pdf', 'page': 0, 'titulo': 'Contenido Web Acerca de Vinculate', 'herramienta': 'ACERCA DE VINCULATE',
'tipo_documento': 'Contenido Web', 'url': 'https://vinculate.concytec.gob.pe/acerca-de-vinculate/'}, page_content='ACERCA DE VINCÚLATE  \nVINCÚLATE
\nPagina web:  https://vinculate.concytec.gob.pe/acerca -de-vinculate/  \nPlataforma de herramientas para la transferencia tecnológica y  \nla vinculación
entre la investigación y la industria  \nCon el objetivo de formar capacidades para la vinculación, estimular la transferencia \ntecnológica y potenciar
que la investigación que se realiza en el país se enfoque en las \nnecesidades de los sectores productivos y de la sociedad, esta plataforma pone a
disposición \nde los investigadores, tecnólogos, innovadores, gestores y emprendedores las siguientes \nherramientas:  \nFormatos de convenios y contratos
para proyectos de I+D+i  \nComprende una guía que facilita el entendimiento y el establecimiento de relaciones \ncontractuales con potenciales socios y
aliados en los procesos de I+D+i para llevar a cabo la \ntransferencia tecnológica de una forma trazable y válida en el sistema legal p eruano. Estos
\nformatos son puestos a disposición del público en general a través de la descarga directa en la \nplataforma con una licencia de libre uso y adaptación.
Los archivos editables se podrán \ndescargar directamente en la web o asesorarse en la elecci ón del formato más adecuado con la \nguía de decisión
automatizada que asistirá a los usuarios en la elección del documento. Visita \nesta sección para mayor información.  \nGuía Metodológica Tangüis (formatos
de convenios y contratos para proyectos de I+D+i)  \nDocumento técnico descargable que facilita el entendimiento y el establecimiento de \nrelaciones entre
potenciales socios (universidades, centros de investigación, institutos públicos \nde investigación en el sector empresarial) a través de formatos
editables de convenios, \ncontratos, cláusulas y acuerdos. Conociendo la necesidad de poder apoyar a los usuarios a \nelegir el documento que se adapte más
a sus necesidades, se ha programado una guía de \ndecisión automatizada que asistirá a los usuarios en la elección d el documento más adecuado. \nDescarga
la Guía Metodológica Tangüis aquí  \nGuía de licenciamiento y negociación  \nDocumento con directrices, mejores prácticas y formatos editables, así como
contratos de \nlicenciamiento para la comercialización de tecnologías entre laboratorios y la industria del \nmismo modo se compartirán estrategias sobre
como llevar a cabo una negocia ción efectiva, \npara lograr que los resultados de investigación lleguen al mercado y proporcionen beneficios a \ntodos los
participantes de este proceso.  \nNiveles de madurez tecnológica (Techonology Readiness Level o TRL)  \nEsta herramienta proporciona conceptos e
información de utilidad para clasificar el nivel de \nmadurez tecnológica de una investigación en su camino al mercado. La propuesta realizada \npara el
SINACYT fue adaptada de los niveles originales propuestos por la N ASA, el Programa \nMarco de Investigación e Innovación de la Unión Europea Horizonte
2020, entre otros. A fin de \nfacilitar su uso, se ha incluido un servicio de calculadora automatizada en línea para asistir a los \nequipos técnicos en la
clasificación de sus investigaciones y desarrollos. Visita esta sección para \nmayor información.  ')
```

Este procesamiento contiene el contenido de una página de un archivo PDF y los metadatos correspondientes como el título, herramienta y tipo de documento, esto facilitará la recuperación de la información en etapas posteriores del desarrollo del asistente virtual.

4.2.4. División en fragmentos

Este paso consistió en dividir los Documents en fragmentos o también conocidos como 'chunks', esta división es crucial para mejorar la eficiencia y precisión del sistema de recuperación de información, ya que permite que los modelos de lenguaje trabajen con segmentos de texto más manejables. Utilizamos el RecursiveCharacterTextSplitter[2] de la librería LangChain para esta tarea.

El RecursiveCharacterTextSplitter puede dividir texto genérico en fragmentos de tamaño controlado, asegurándose de que cada fragmento tenga suficiente contexto para ser útil en las tareas de generación y recuperación de información. Esta técnica, ilustrada en la Figura 19 muestra los principales parámetros que se configuran chunk_size y chunk_overlap.

Figura 19

División de documentos en fragmentos

```
[ ]  #Este divisor de texto es el recomendado para texto genérico
     from langchain.text_splitter import RecursiveCharacterTextSplitter

     # AjustaR
     chunk_size = 2000
     chunk_overlap = 250

     # Dividir los documentos en chunks
     text_splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size, chunk_overlap=chunk_overlap)
     splits = text_splitter.split_documents(documents)
```

A continuación, se indica la configuración de los parámetros.

- chunk_size: Define el número máximo de caracteres en cada fragmento. En este caso, se ha establecido en 2000 caracteres.

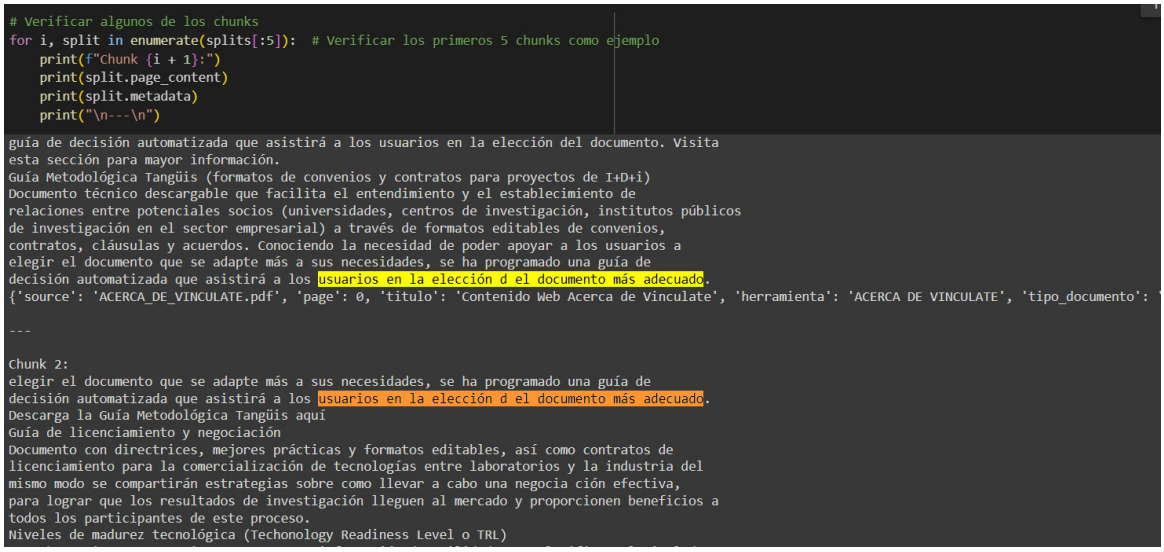
[2]Para obtener más información sobre el proceso de division de documentos, consulte https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/recursive_text_splitter/

- chunk_overlap: Define el número de caracteres que se solapan entre fragmentos consecutivos. En este caso, se ha establecido en 250 caracteres. Este solapamiento ayuda a mantener la coherencia y el contexto entre los fragmentos.

Si los fragmentos son demasiado pequeños, pueden perder contexto; si son demasiado grandes, pueden ser difíciles de manejar y procesar para los modelos de lenguaje, también es importante usar el solapamiento para que no se pierda contexto entre los bordes de los fragmentos. En la Figura 20, se puede visualizar un ejemplo de los fragmentos generados, donde se observa cómo se mantiene el contexto a pesar de la división.

Figura 20

Visualización de los fragmentos generados



4.3. Generación de Embeddings

Esta etapa es esencial en el proceso de datos para sistemas de Recuperación Aumentada por Generación (RAG). Los embeddings convierten el texto en representaciones numéricas vectoriales de alta dimensión que son más fáciles de procesar por los modelos de lenguaje. Para este trabajo, se utilizó el modelo text-embedding-3-small de OpenAI, como se visualiza en la Figura 21, que proporciona un alto rendimiento y se integra bien en nuestro sistema.

A continuación, inicializaremos la clase OpenAIEmbeddings de Langchain que es una clase diseñada para interactuar con modelos de embeddings

Figura 21

Instanciado la clase OpenAIEmbedding

```
from langchain_openai import OpenAIEmbeddings
#ahora inicilaizaremos con el modelo text-embedding-3-small
#Mayor rendimiento en comparación con el modelo de integración ADA de segunda generación
embeddings = OpenAIEmbeddings(model="text-embedding-3-small")
```

Para ilustrar cómo funciona, esta clase se utilizó un texto de prueba y se hizo uso de la clase, en la Figura 22 se puede observar su representación vectorial del texto ingresado, se muestra solo los 5 primeros valores de un vector de dimensión 1536.

Figura 22

Visualización de una representación vectorial

```
# ejemplo de como funciona el modelo de incrustaciones
text = "Este es un documento de prueba"

[ ] #visualizamos una muestra de los valores vectoriales
query_result = embeddings.embed_query(text)
query_result[:5]

[0.04074642062187195,
 0.03165924921631813,
 -0.009721027687191963,
 -0.002275999402627349,
 -0.02103511430323124]

[ ] #aqui validamos la dimensión del vector
len(query_result)

1536
```

Luego de que se instanció la clase en la variable embeddings esta será usada más adelante por otra herramienta de Langchain.

4.4. Creación de la base de datos Vectorial

En esta etapa se construyó la base de datos vectorial, utilizando Chroma, esta base de datos permite almacenar los embeddings generados a partir de los chunks de los

documentos, lo cual facilita la búsqueda eficiente y las comparaciones por similitud que son características fundamentales de este tipo de bases.

Como primer paso para la creación de esta base es definir un directorio donde se almacenarán los datos de la base de datos vectorial. Este directorio actúa como almacenamiento persistente para los embedding generados.

Figura 23

Generación de la base de datos vectorial

```
# Definimos el directorio de persistencia
# Donde se almacenarán nuestra base de datos Vectorial
persist_directory = "/content/embedding_data"
vectorstore = Chroma.from_documents(documents=splits, embedding=embeddings, persist_directory=persist_directory)
```

En la Figura 23 se observa el método Chroma.from_documents[3] el cual convierte los fragmentos de los documentos (chunks) que hemos generado con anterioridad a vectores mediante el modelo de embeddings definido también con anterioridad el cual fue el modelo text-embedding-3-small de OpenAI. Estos vectores son almacenados en Chroma, creando así una representación de los documentos en un espacio vectorial, dicha representación facilita la búsqueda y las comparaciones de similitud entre los diferentes fragmentos.

Para ilustrar la funcionalidad de la base de datos vectorial, realizamos una búsqueda de vectores por similitud utilizando una consulta de prueba. Configuramos k, el número de resultados a retornar y los resultados obtenidos se muestran en la Figura 24, donde se puede apreciar cómo se visualizan los metadatos de los documentos más relevantes, facilitando así la interpretación de los resultados.

[3] Para obtener más información sobre la herramienta para creación de base de datos vectoriales Chroma ofrecido por Langchain visite
https://python.langchain.com/v0.2/docs/integrations/vectorstores/chroma/

Figura 24

Visualización de metadatos de documentos relevantes recuperados

```
#TEST de la base de datos
query = "Que es la calculadora de madurez"
k = 10
results = vectorstore.similarity_search_with_relevance_scores(query, k=k)
# imprimimos pagina y titulo del documento, mas similaridad
[(r[0].metadata['page'],r[0].metadata['source'], r[1]) for r in results]
```

```
[(7, 'NIVELES_DE_MADUREZ_TECNOLOGICA.pdf', 0.36115465667162316),
 (7, 'NIVELES_DE_MADUREZ_TECNOLOGICA.pdf', 0.3128122123567991),
 (0, 'NIVELES_DE_MADUREZ_TECNOLOGICA.pdf', 0.17552250649292078),
 (9, 'RP-020-2022-CONCYTEC-P.pdf', 0.16921775942417616),
 (5, 'RP-020-2022-CONCYTEC-P.pdf', 0.164980736743376),
 (7, 'RP-020-2022-CONCYTEC-P.pdf', 0.13307515095214406),
 (12, 'RP-020-2022-CONCYTEC-P.pdf', 0.12445603613796918),
 (13, 'RP-020-2022-CONCYTEC-P.pdf', 0.11360676298168892),
 (17, 'RP-020-2022-CONCYTEC-P.pdf', 0.11200568850045711),
 (6, 'RP-020-2022-CONCYTEC-P.pdf', 0.10978438099653898)]
```

En el ejemplo anterior, la consulta (query) se convierte en un vector utilizando la misma función de embeddings, y se compara con los vectores almacenados en la base de datos. Los resultados se ordenan según su relevancia, permitiendo recuperar los fragmentos más similares y relevantes para la consulta, podemos observar los documentos relacionados a los Niveles de Madurez Tecnológica que se calculan en base a la calculadora, también se visualiza la Directiva del TRL lo que demuestra que este método permite la recuperación rápida y precisa de información relevante, crucial en la implementación de sistemas RAG.

Finalmente se obtuvo la base de datos vectorial generada y se procedió a descargar en un comprimido el directorio persistente que fue definido al inicio, este directorio contiene archivos binarios que pueden ser importadas fácilmente.

4.5. Implementación del recuperador

En esta etapa nos corresponde añadir a nuestro sistema un método que permite buscar y recuperar documentos relevantes en respuesta a consultas de los usuarios, utilizando los embeddings almacenados en la base de datos vectorial.

Para crear el recuperador o retriever de documentos, utilizamos el vectorstore[4] de Chroma previamente creado. Configuramos el número de resultados k que deseamos

[4] Para más información de recuperadores con Langchain visite el siguiente enlace https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/vectorstore/

recuperar por cada consulta. A continuación, se muestra en la figura 25 cómo inicializar el recuperador.

Figura 25

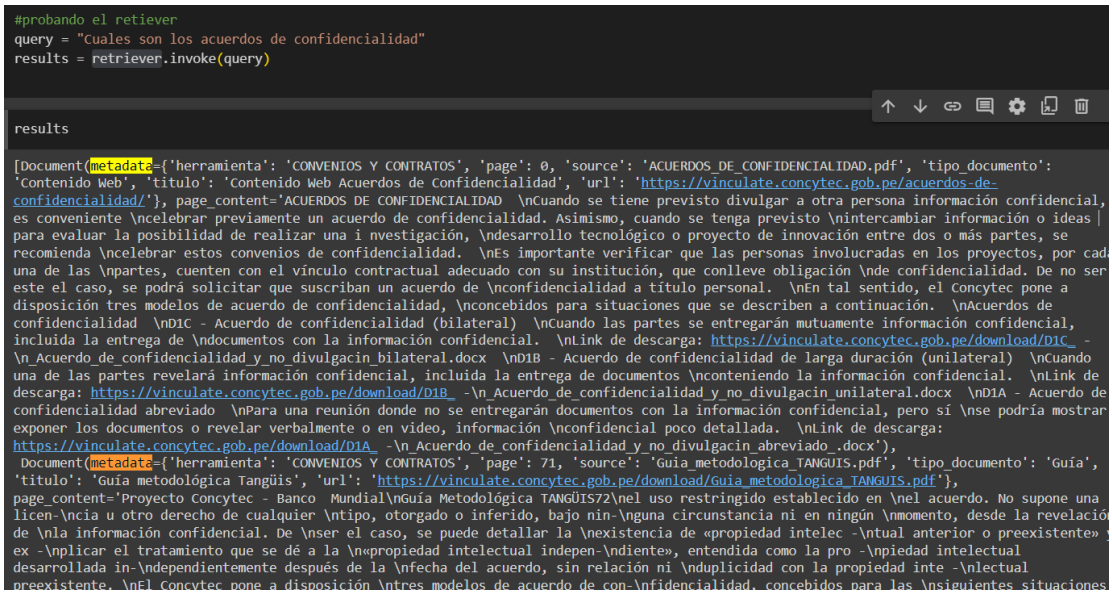
Implementación del Recuperador

```
#generamos nuestro retriever(recuperador de documentos)
k = 10
retriever = vectorstore.as_retriever(search_kwargs={"k":k})
```

Se verificó el correcto funcionamiento del recuperador que se mostrará en el siguiente ejemplo, en el que la consulta busca documentos relevantes que contienen información sobre "acuerdos de confidencialidad", en la Figura 26 se puede observar los resultados proporcionados por el recuperador.

Figura 26

Documentos recuperados y sus metadatos



Cada documento recuperado incluye metadatos como la herramienta a la que pertenece, la página, la fuente, el título y la URL de descarga. A continuación.

Finalmente, el sistema de recuperación utiliza los embeddings de los documentos para encontrar aquellos que son más relevantes en base a la similitud con la consulta del usuario. Este enfoque asegura que las respuestas proporcionadas por el asistente virtual

sean precisas y contextualmente relevantes. Al configurar parámetros como k, podemos controlar la cantidad de resultados a mostrar, optimizando la relevancia y precisión de las respuestas

4.6. Generación de respuestas

La generación de respuestas es una de las etapas más importantes en el desarrollo de un asistente virtual basado en (RAG). En esta estapa, el sistema utiliza los grandes modelos de lenguaje (LLM) para proporcionar respuestas precisas y contextualmente relevantes a las consultas de los usuarios, integrando la información recuperada de la base de datos vectorial.

Como primer paso en la generación de respuestas se define el modelo de lenguaje que se utilizará. En este trabajo, se utilizó el modelo gpt-3.5-turbo de OpenAI. Utilizaremos la clase ChatOpenAI de Langchain que nos permite generar interacciones de chat con el api de OpenAI, La Figura 27 muestra el proceso de instanciar la clase con una temperatura de 0.9 para ajustar la creatividad de las respuestas y con el parámetro verbose habilitado para obtener información detallada durante la ejecución.

Figura 27

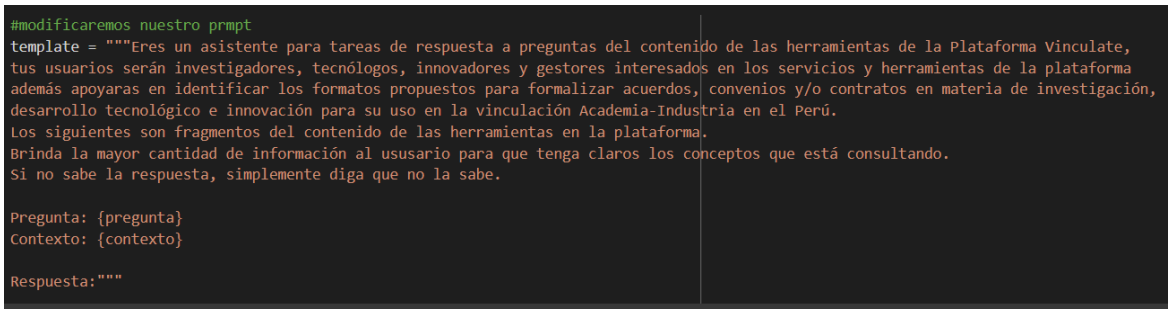
Instanciando el modelo de lenguaje

```
# Definir el modelo de lenguaje
llm=ChatOpenAI(temperature=0.9,chat_model_name='gpt-3.5-turbo',verbose=True)
```

Luego se define el template del prompt en el que se indica como se estructurarán las consultas y el contexto antes de ser enviados al modelo de lenguaje. En la Figura 28 se configura un prompt específico para el asistente virtual de Vincúlate, proporcionando detalles sobre la función del asistente y cómo debe manejar las preguntas.

Figura 28

Template de instrucciones del asistente



```
#modificaremos nuestro prmpt
template = """Eres un asistente para tareas de respuesta a preguntas del contenido de las herramientas de la Plataforma Vinculate,
tus usuarios serán investigadores, tecnólogos, innovadores y gestores interesados en los servicios y herramientas de la plataforma
además apoyaras en identificar los formatos propuestos para formalizar acuerdos, convenios y/o contratos en materia de investigación,
desarrollo tecnológico e innovación para su uso en la vinculación Academia-Industria en el Perú.
Los siguientes son fragmentos del contenido de las herramientas en la plataforma.
Brinda la mayor cantidad de información al ususario para que tenga claros los conceptos que está consultando.
Si no sabe la respuesta, simplemente diga que no la sabe.

Pregunta: {pregunta}
Contexto: {contexto}

Respuesta:"""
```

El siguiente paso consiste en definir la cadena RAG, la cual se configura con los componentes necesarios para recuperar el contexto adecuado y generar respuestas coherentes. Esta cadena, tal como se muestra en la Figura 29, abarca los siguientes pasos clave:

1. **Retriever**: Este componente se encarga de localizar y recuperar los documentos más relevantes desde la base de datos vectorial en función de la consulta proporcionada por el usuario. Utiliza técnicas avanzadas de búsqueda y recuperación de información para garantizar que los documentos seleccionados estén alineados con las necesidades específicas del usuario.
2. **Format Docs**: Una vez que los documentos relevantes han sido recuperados, este proceso se encarga de formatearlos adecuadamente para su utilización dentro del contexto del prompt. Esta etapa asegura que los documentos estén en el formato correcto y sean fácilmente integrables en la construcción del contexto requerido para la generación de respuestas.
3. **Prompt**: Utiliza el template definido para estructurar la consulta y el contexto.
4. **LLM**: El modelo de lenguaje de aprendizaje profundo utiliza el contexto y la consulta estructurada para generar respuestas. Este componente se basa en técnicas de procesamiento de lenguaje natural (NLP) avanzadas que permiten la comprensión semántica del contexto y la producción de respuestas que son relevantes y contextualmente apropiadas.
5. **Output Parser**: Una vez que la respuesta ha sido generada por el modelo de lenguaje, este componente se encarga de analizar y formatear la salida para ser

entregada al usuario. Este paso es fundamental para asegurar que la respuesta proporcionada sea clara, estructurada y directamente útil para el usuario final.

Figura 29

Creación de cadena Rag con Langchain

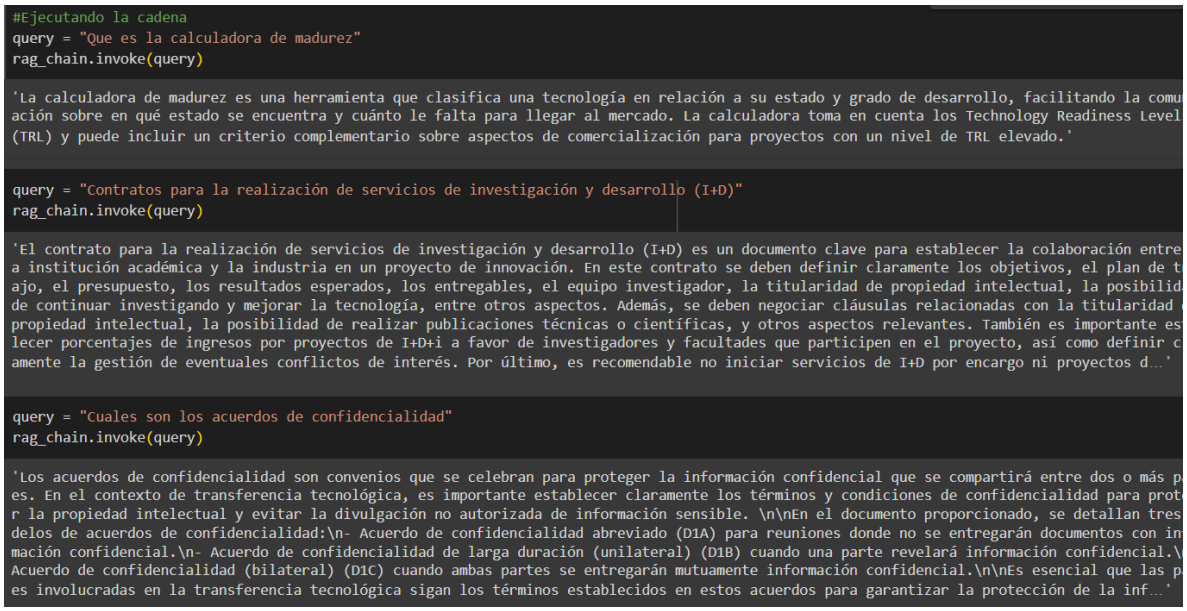
```
#Pipeline para la cadena RAG

from langchain_core.runnables import RunnablePassthrough
# Creamos nuestra cadena RAG
rag_chain = (
    {"contexto": retriever | format_docs, "pregunta": RunnablePassthrough()}
    | prompt
    | llm
    | StrOutputParser())
```

Para validar el funcionamiento de esta cadena, se ejecutan consultas de ejemplo para verificar que las respuestas sean precisas y contextualmente adecuadas. Los resultados obtenidos, presentados en la Figura 30, ilustran la capacidad del sistema para generar respuestas detalladas y relevantes utilizando la información almacenada en la base de datos vectorial y procesada por el modelo de lenguaje.

Figura 30

Generación de respuestas a consultas con RAG



El procedimiento de implementación basado en RAG descrito anteriormente ilustra cómo la recuperación aumentada por generación, en combinación con LangChain, puede integrarse efectivamente en el marco de asistentes virtuales con IA generativa. Este enfoque integral cubre la recopilación de datos, el procesamiento, la generación de embeddings, la búsqueda y la generación de respuestas, asegurando que el asistente proporcione respuestas precisas y contextualmente relevantes a las consultas de los usuarios de la plataforma.

4.7. Interfaz de Usuario

Se implementó una interfaz de usuario intuitiva para facilitar la interacción de los usuarios con el asistente virtual, para este trabajo se utilizó el framework de Gradio, que permite crear interfaces web de manera rápida y sencilla con especial énfasis en aplicaciones de aprendizaje automático.

A continuación, se define la estructura de la interfaz utilizando los bloques de Gradio (Blocks). En la Figura 31, se puede observar cómo se han dispuesto estos bloques, incluyendo una tabla para visualizar la base de datos de documentos y un chatbot para interactuar con el asistente virtual.

Figura 31

Bloques en la interfaz de Gradio

```
import gradio as gr
with gr.Blocks(theme=gr.themes.Soft()) as demo:
    gr.Markdown("""<h1 style="text-align: center;">Asistente de Herramientas de la Plataforma VINCULATE</h1>""")
    gr.Markdown("""Este asistente virtual utiliza Recuperación Aumentada por Generación (RAG) y el modelo de lenguaje GPT-3.5 Turbo para respond
    with gr.Row():
        with gr.Column(scale=1):
            gr.Markdown("## Base de documentos")
            table = gr.DataFrame(value=metadata)

        with gr.Column(scale=4):
            gr.Markdown("## Chatbot")
            chatbot = gr.Chatbot(height=450)
            msg = gr.Textbox(placeholder="Realiza una pregunta sobre las herramientas de Vinculate", show_label=False)
            submit_btn = gr.Button("Consultar", variant="primary")
            msg.submit(respond, [msg, chatbot], [msg, chatbot])
            submit_btn.click(respond, [msg, chatbot], [msg, chatbot])
            clear = gr.ClearButton([msg, chatbot])

            gr.Markdown("### Ejemplos")
            examples = gr.Examples(examples=["Que es la guia Tanguis", "Cuales son los acuerdos de confidencialidad", "Como puedo negociar una t

demo.launch(auth=authenticate)
```

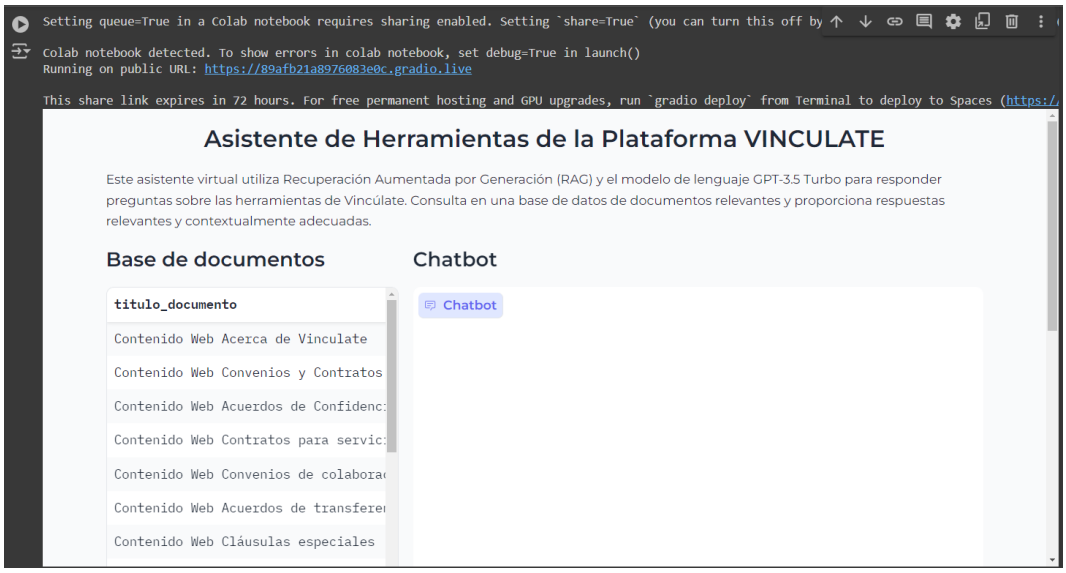
Se detallan las componentes principales de la interfaz.

1. Base de Documentos: Se utilizó un componente DataFrame para mostrar la base de datos de documentos que hemos cargado previamente. Esto permite a los usuarios visualizar los documentos disponibles en la plataforma.
2. Chatbot: Creamos un componente Chatbot que permite a los usuarios interactuar directamente con el asistente virtual. Este componente incluye una caja de texto (Textbox) para que los usuarios ingresen sus preguntas, un botón de envío (Button) para enviar las preguntas y un botón de limpieza (ClearButton) para reiniciar el chat.
3. Ejemplos de Preguntas: Se añadió un componente Examples que muestra ejemplos de preguntas que los usuarios pueden realizar. Estos ejemplos ayudan a los usuarios a entender el tipo de consultas que pueden hacerle al asistente virtual.

Para validar la funcionalidad de la interfaz, se implementó y probó en un entorno de notebook. Como se muestra en la Figura 32, la interfaz permite a los usuarios visualizar la base de datos de documentos y formular consultas al asistente virtual de manera intuitiva y eficiente.

Figura 32

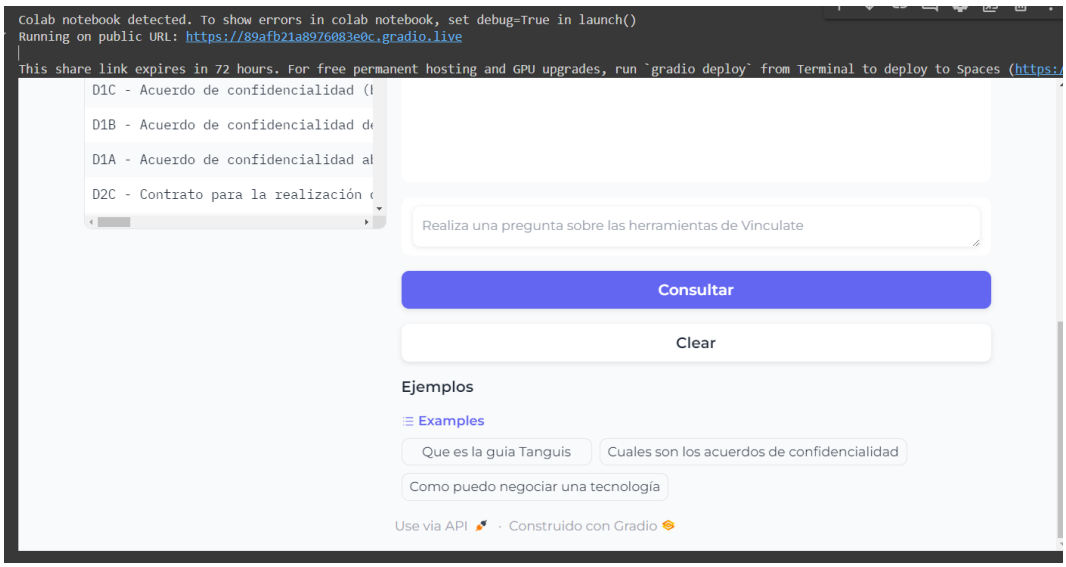
Visualización de la interfaz en notebook



Además, para mejorar la interacción del usuario, se añadieron botones que facilitan la navegación y el uso del asistente. Estos botones, que incluyen opciones para enviar preguntas, limpiar el chat, y seleccionar ejemplos de consultas, son claramente visibles en la Figura 33. Estos elementos de diseño garantizan que la experiencia del usuario sea lo más fluida y amigable posible.

Figura 33

Botones de la interfaz



Además, se agregó una ventana de autenticación para asegurar que solo usuarios autenticados puedan acceder a la interfaz, esta ventana se puede observar en la Figura 36.

4.8. Despliegue del sistema en HugginFace

En esta sección, se detallará el proceso de despliegue del asistente virtual para la plataforma Vincúlate en Hugging Face Spaces, el cual proporciona una plataforma flexible y fácil de usar para alojar aplicaciones de IA generativa.

En etapas anteriores hemos visto el flujo detallado desde la generación y procesamiento de los datos hasta la generación de las respuestas en base al contexto, sin embargo, como se puede observar este código ha estado seccionado y no es ordenado para un entorno de desarrollo por ello se crea el archivo app.py, el código fuente puede revisar el Anexo 2, este archivo se creó para unificar y organizar todo el código necesario para implementar el asistente virtual en un solo lugar. Este enfoque facilita el despliegue y

mantenimiento del sistema en Hugging Face Spaces. Al consolidar todas las dependencias, configuraciones y funciones en un archivo, se asegura que el asistente virtual sea fácil de ejecutar y gestionar. Además, Hugging Face Spaces requiere un archivo principal que defina la aplicación para ejecutarla correctamente en su entorno, además Gradio también necesita de una sola función para ejecutarse en el chatbot.

Figura 34

Clase VinculateBot contenida en app.py

```
class VinculateBot:
    def __init__(self, temp=0.9,
                       chat_model_name='gpt-3.5-turbo',
                       embeddings_model_name='text-embedding-3-small',
                       num_results=10,
                       persist_directory='embedding_data',
                       ):

        self.embeddings_model_name = embeddings_model_name
        self.temp = temp
        self.chat_model_name = chat_model_name
        self.num_results = num_results
        self.persist_directory = persist_directory

        # Embeddings Model
        self.embeddings_model = OpenAIEmbeddings(api_key=api_key, model=self.embeddings_model_name)

        # Initialize Chat Model
        self.chat_model = ChatOpenAI(api_key=api_key, temperature=self.temp, model=self.chat_model_name)

        # Set up vector store and other components
        self.context_retriever = ContextRetriever(embeddings_model=self.embeddings_model, persist_directory=self.persist_directory
        self.chat_history_formatter = ChatHistoryFormatter()
        self.question_contextualizer = QuestionContextualizer(self.chat_model)
        self.answer_generator = AnswerGenerator(self.chat_model)
        self.result_formatter = ResultFormatter()

    def process_chat(self, new_question, chat_history):
        formatted_chat_history = self.chat_history_formatter.format_chat_history(chat_history)
        contextualized_question = self.question_contextualizer.contextualize_question(formatted_chat_history, new_question)
        context = self.context_retriever.get_context(contextualized_question)
        answer = self.answer_generator.generate_answer(context=context, question=contextualized_question)
        final_result = self.result_formatter.format_result(answer, context)
        return final_result
```

Para consolidar todos los componentes necesarios para el funcionamiento del asistente virtual, se ha desarrollado la clase VinculateBot, que se encuentra contenida en el archivo app.py. Esta clase, ilustrada en la Figura 34, encapsula la configuración del modelo de lenguaje, el manejo de embeddings, la recuperación de contexto, y la generación de respuestas.

A continuación, se describe el contenido de app.py

1. Importación de Librerías: Se importan todas las librerías necesarias, incluyendo Gradio para la interfaz de usuario, Pandas para la manipulación de

datos, y LangChain para la implementación del modelo de lenguaje y la base de datos vectorial.

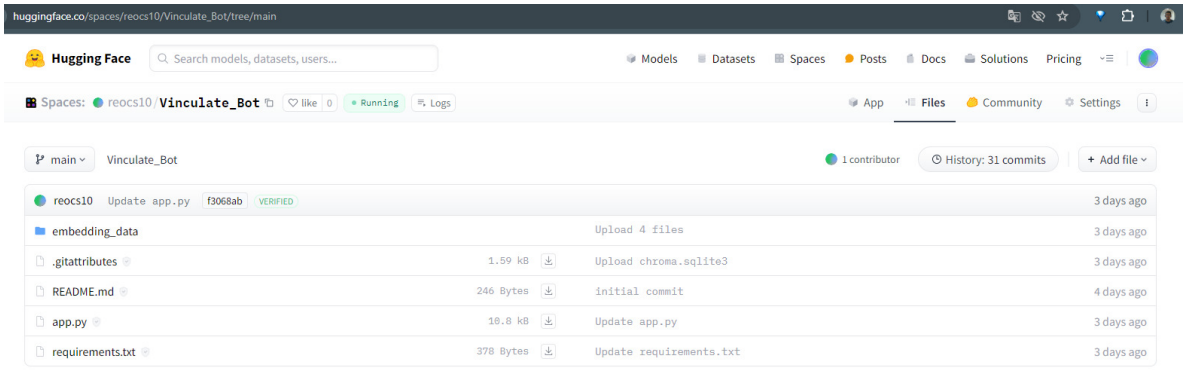
2. Configuración de Credenciales: Se cargan las variables de entorno y se configuran las credenciales de OpenAI para acceder a los modelos de lenguaje.
3. Definición de Clases: Se definieron varias clases para manejar la recuperación de contexto, la generación de respuestas y el formato de los resultados. Estas clases encapsulan la funcionalidad necesaria para cada etapa del proceso de generación de respuestas. Estas son:
    - ContextRetriever: Recupera el contexto relevante desde la base de datos vectorial.
    - ChatHistoryFormatter: Formatea el historial de chat para ser utilizado en la generación de respuestas.
    - QuestionContextualizer: Reformula las preguntas del usuario utilizando el historial de chat.
    - AnswerGenerator: Genera respuestas utilizando el modelo de lenguaje.
    - ResultFormatter: Formatea los resultados para ser presentados al usuario, se incluyen los metadatos y links relevantes.
4. Se define una clase VinculateBot que integra todas las clases anteriores y maneja el flujo completo desde la recepción de una pregunta hasta la generación de la respuesta.
5. Se configura la interfaz de usuario utilizando Gradio, incluyendo componentes para autenticación, chat y ejemplos de preguntas.

También se creó un archivo requirements.txt para listar todas las dependencias necesarias para la aplicación. Hugging Face Spaces utilizará este archivo para instalar las dependencias.

Recordemos que anteriormente se había descargado la base de datos vectorial en ficheros binarios, la cual fue subida en el repositorio de Huggin Face Space junto con el fichero app.py y reqirements.txt, como se puede observar en la Figura 35.

Figura 35

Repositorio de Código Fuente en Huggin Face



Una vez subidos los archivos, Hugging Face Spaces automáticamente instalará las dependencias y ejecutará la aplicación. Luego se puede verificar el estado y acceder a la URL proporcionada para el espacio.

Finalmente podemos observar en la Figura 36 y Figura 37, el asistente virtual completamente desplegado y disponible para los usuarios de la plataforma Vincúlate, proporcionando respuestas precisas y contextualmente relevantes a sus consultas.

Figura 36

Ventana del Login del asistente desplegado en Spaces

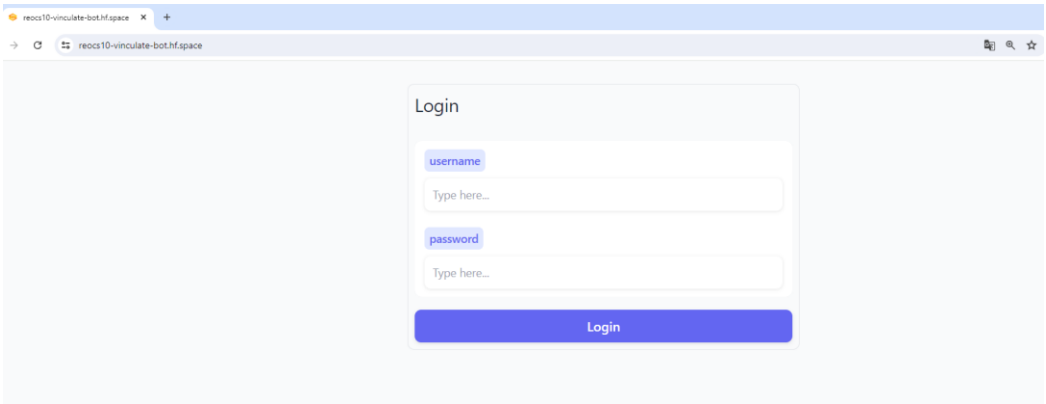
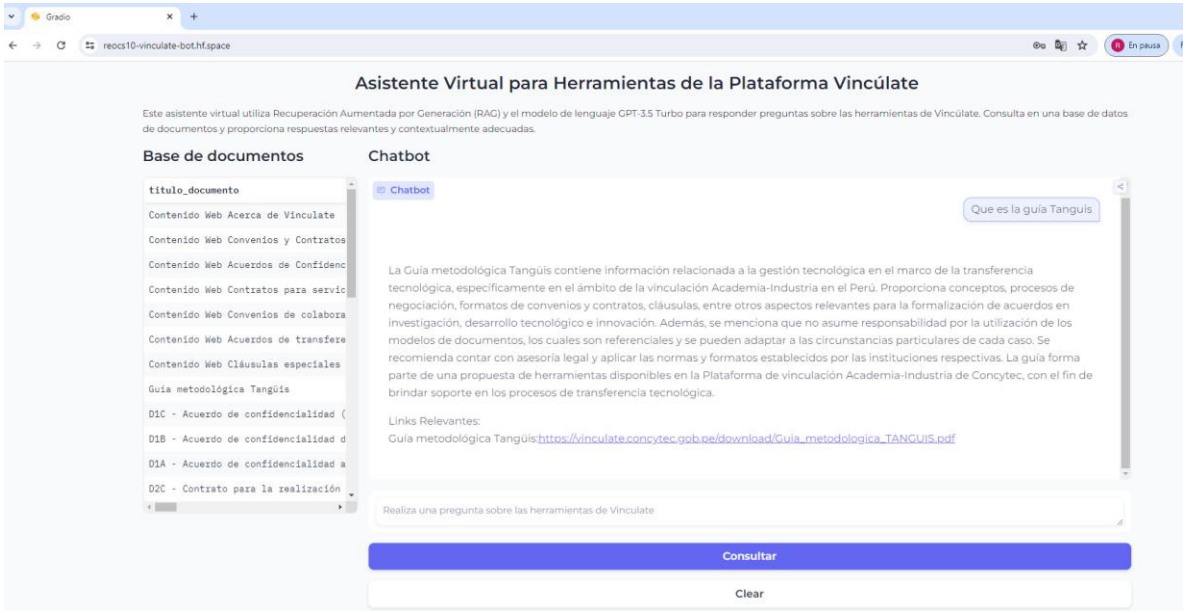


Figura 37

Ventana principal del asistente virtual desplegado en Spaces



4.9. Evaluación

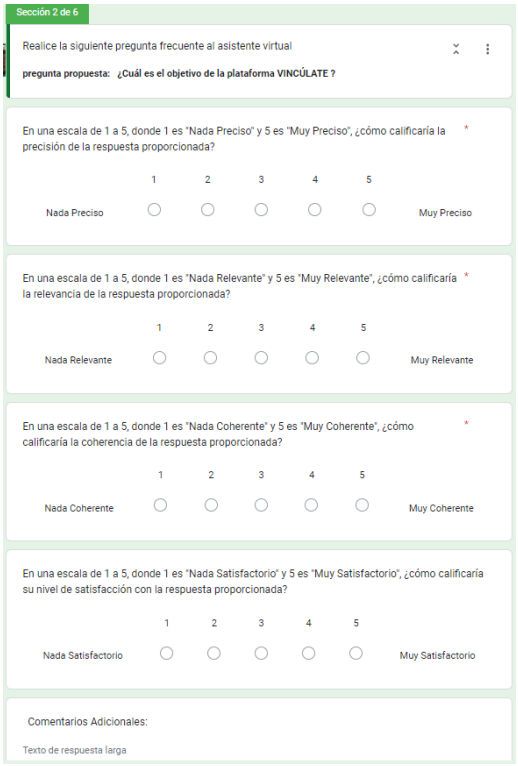
Con el objetivo de evaluar la eficacia y precisión del asistente virtual de la plataforma Vincúlate, se llevó a cabo una encuesta dirigida a los usuarios de la plataforma. La encuesta, cuya estructura se puede observar en la Figura 38, fue diseñada para recolectar opiniones y valoraciones sobre la calidad de las respuestas proporcionadas por el asistente virtual, considerando aspectos como la precisión, relevancia, coherencia y nivel de satisfacción.

Las preguntas propuestas para el análisis fueron las siguientes:

1. ¿Cuál es el objetivo de la plataforma VINCÚLATE?
2. ¿Qué herramientas encuentro en la Plataforma VINCÚLATE?
3. ¿Por qué Concytec utiliza los TRL para clasificar mi tecnología?
4. ¿Los documentos que encuentro en la plataforma Vincúlate son instrumentos legales?

Figura 38

Encuesta compartida con usuarios de la plataforma.



A cada uno de los participantes se les pidió que formularan una pregunta al asistente virtual y evaluaran la respuesta proporcionada utilizando una escala de 1 a 5, donde 1 es "Nada Preciso/Relevante/Coherente/Satisfactorio" y 5 es "Muy Preciso/Relevante/Coherente/Satisfactorio". La estructura de la encuesta se detalla a continuación:

- ¿Cómo calificaría la precisión de la respuesta proporcionada?
  (1: Nada Preciso, 5: Muy Preciso)
- ¿Cómo calificaría la relevancia de la respuesta proporcionada?
  (1: Nada Relevante, 5: Muy Relevante)
- ¿Cómo calificaría la coherencia de la respuesta proporcionada?
  (1: Nada Coherente, 5: Muy Coherente)
- ¿Cómo calificaría su nivel de satisfacción con la respuesta proporcionada?
  (1: Nada Satisfactorio, 5: Muy Satisfactorio)

La información de las encuestas fue analizada principalmente mediante técnicas de análisis cuantitativo debido a la cantidad de respuestas recibidas. Las respuestas estructuradas fueron procesadas utilizando estadísticas descriptivas para evaluar la precisión, relevancia, coherencia y satisfacción con las respuestas del asistente virtual. Se calcularon medidas como el promedio para proporcionar una visión del desempeño del asistente en cada uno de los aspectos evaluados.

Adicionalmente, se realizó una lectura de las respuestas abiertas para obtener una comprensión general de las percepciones y comentarios cualitativos de los usuarios. Los datos recolectados proporcionan una visión integral sobre cómo los usuarios perciben el desempeño del asistente con relación a sus expectativas y necesidades.

CAPÍTULO V: RESULTADOS

En este capítulo se presentarán los hallazgos derivados de la encuesta dirigida a los usuarios de la plataforma Vincúlate, quienes evaluaron las respuestas generadas por el asistente en términos de:

- Precisión de las Respuestas: Se evaluó la capacidad del asistente para proporcionar respuestas precisas a las preguntas de los usuarios basadas en la información de los documentos de la plataforma.
- Relevancia de la Información: Se analizó si las respuestas generadas eran relevantes para las consultas realizadas por los usuarios.
- Coherencia del Contenido: Se revisó la coherencia y fluidez de las respuestas proporcionadas por el asistente.
- Satisfacción del Usuario: Se midió el nivel de satisfacción de los usuarios con las respuestas obtenidas mediante encuestas.

Los datos recolectados fueron analizados cuantitativamente y fue resumido y presentado en las Tablas 1 y 2. Estas tablas muestran los puntajes promedio obtenidos para cada pregunta y para cada dimensión evaluada.

Tabla 1

Resumen Promedios de Puntajes por Pregunta

Pregunta	Precisión	Relevancia	Coherencia	Satisfacción
¿Cuál es el objetivo de la plataforma VINCÚLATE?	4.2	4.3	4.1	4.2
¿Qué herramientas encuentro en la Plataforma VINCÚLATE?	4	4.1	4	4.1
¿Por qué Concytec utiliza los TRL para clasificar mi tecnología?	3.8	3.9	3.7	3.8

¿Los documentos que encuentro en la plataforma Vincúlate son instrumentos legales?	4.1	4.2	4	4.1
Pregunta Libre	4	4.2	4.1	4.2

A continuación, se presenta un análisis detallado para la Tabla 1, podemos interpretar los siguientes resultados:

- Los puntajes indican que las respuestas sobre el objetivo de la plataforma son altamente precisas, relevantes y coherentes, resultando en una alta satisfacción por parte de los usuarios.
- Las respuestas sobre las herramientas disponibles en la plataforma son consistentes y bien valoradas, aunque hay un ligero margen de mejora en la precisión y la coherencia.
- Las respuestas sobre la naturaleza legal de los documentos en la plataforma son bien valoradas, aunque hay espacio para mejorar la coherencia.
- Las respuestas a preguntas libres también fueron bien valoradas, mostrando que el asistente puede manejar una variedad de consultas de manera efectiva.

Tabla 2

Resumen Puntaje Promedio por dimensión

Dimensión	Promedio
Precisión	4.0
Relevancia	4.1
Coherencia	4.0
Satisfacción	4.1

La Tabla 2 presenta los puntajes promedio agregados por cada dimensión evaluada. Los resultados son los siguientes:

- El promedio de precisión fue de 4.0, indicando que, en general, el asistente virtual proporciona respuestas precisas a las consultas de los usuarios.
- La relevancia obtuvo un promedio de 4.1, lo que sugiere que las respuestas son, en su mayoría, relevantes y útiles para los usuarios.
- La coherencia también tuvo un promedio de 4.0, demostrando que las respuestas son lógicas y bien estructuradas.
- La satisfacción de los usuarios con las respuestas proporcionadas tuvo un promedio de 4.1, indicando un alto nivel de aceptación y satisfacción con el desempeño del asistente virtual.

Además del análisis cuantitativo, se realizó un análisis cualitativo de las respuestas abiertas de los usuarios. Se observó que:

- Los usuarios valoraron positivamente la capacidad del asistente para proporcionar información contextualizada y detallada.
- Algunos usuarios sugirieron mejoras en la interfaz de usuario para hacerla más intuitiva.
- Hubo comentarios positivos sobre la velocidad y eficiencia del asistente al generar respuestas.

Los resultados obtenidos indican que el asistente virtual para la plataforma Vincúlate proporciona respuestas precisas y relevantes, mejorando significativamente el acceso a la información para los usuarios de la plataforma. Además, el sistema de recuperación aumentada por generación (RAG) y el modelo de lenguaje GPT-3.5 turbo mostraron una capacidad robusta para buscar y recuperar documentos relevantes en respuesta a las consultas de los usuarios, utilizando embeddings almacenados en la base de datos vectorial creada. Este enfoque asegura que las respuestas proporcionadas sean precisas y contextualmente relevantes.

Al comparar estos resultados con la teoría existente y trabajos previos, se puede observar que el uso de modelos de lenguaje avanzados y técnicas de RAG mejora efectivamente la calidad de las respuestas en asistentes virtuales. Estos hallazgos son consistentes con estudios previos que destacan la capacidad de los grandes modelos de lenguaje para comprender y generar respuestas contextualmente adecuadas.

CAPÍTULO VI: CONCLUSIONES y RECOMENDACIONES

CONCLUSIONES

La presente investigación ha demostrado con éxito la viabilidad y efectividad de implementar un asistente virtual basado en inteligencia artificial generativa para la plataforma Vincúlate del Concytec. Este desarrollo ha logrado mejorar significativamente el acceso a la información, proporcionando respuestas precisas y relevantes a las consultas de los usuarios. El uso de técnicas avanzadas de Recuperación Aumentada por Generación (RAG) y grandes modelos de lenguaje (LLM) como GPT-3.5-turbo ha sido fundamental para alcanzar este nivel de precisión y relevancia en las respuestas generadas.

La evaluación del asistente virtual, realizada a través de encuestas a usuarios frecuentes de la plataforma y análisis de especialistas, ha revelado un alto grado de satisfacción con respecto a la precisión, relevancia, coherencia y utilidad de las respuestas proporcionadas. Se destacó particularmente la capacidad del sistema para comprender las preguntas, buscar información relevante en la base de datos y generar respuestas claras y concisas. La integración de técnicas de RAG ha permitido reducir significativamente la incidencia de errores comunes en los sistemas de IA generativa, como las alucinaciones, mejorando así la capacidad del sistema para manejar preguntas específicas del dominio de Vincúlate.

La implementación de una base de datos vectorial utilizando Chroma, en conjunto con el uso del framework Langchain para orquestar la generación de respuestas, ha permitido al asistente proporcionar un soporte técnico altamente eficiente. Este enfoque ha optimizado la accesibilidad y la coherencia del contenido proporcionado, superando las dificultades asociadas con la navegación a través de grandes volúmenes de documentos, un desafío común entre los usuarios de la plataforma Vincúlate.

El despliegue exitoso del asistente virtual en Hugging Face Spaces ha demostrado la viabilidad de implementar soluciones de IA generativa accesibles y escalables para mejorar los servicios de información en plataformas como Vincúlate. Esta elección de despliegue facilita el acceso y uso del asistente por parte de una comunidad más amplia, contribuyendo así a la democratización de las herramientas de IA en el ámbito académico y científico.

Es importante destacar que, aunque el asistente virtual está actualmente desplegado solo en Hugging Face y está pendiente su implementación directa dentro de la plataforma Vincúlate, su desarrollo y evaluación han sentado las bases para la aplicación futura de tecnologías de IA generativa en el ecosistema de ciencia y tecnología en Perú. Incluso en su estado actual, todas las respuestas generadas por el asistente conducen a enlaces de la plataforma Vincúlate o a documentos que pertenecen a esta, lo que demuestra su capacidad para mejorar el acceso a la información específica de la plataforma.

En conclusión, este trabajo de investigación representa una contribución significativa al campo de la Inteligencia Artificial aplicada a la educación y la transferencia de tecnología. El asistente virtual desarrollado ha demostrado ser replicable, sugiriendo su potencial aplicación en otras bases de conocimiento gestionadas por Concytec. Este proyecto marca un paso importante hacia la digitalización y accesibilidad de la información en el ámbito académico y científico peruano, demostrando el potencial de las tecnologías de inteligencia artificial para transformar la interacción de los usuarios con volúmenes de datos en dominios específicos.

RECOMENDACIONES

Con base en los resultados y conclusiones de esta investigación, se proponen las siguientes recomendaciones para futuros desarrollos y mejoras en el campo de los asistentes virtuales basados en inteligencia artificial generativa, específicamente para la plataforma Vincúlate del Concytec.

En primer lugar, se recomienda la implementación completa del asistente virtual dentro de la plataforma Vincúlate. Aunque el asistente ha sido desplegado con éxito en Hugging Face Spaces, su integración directa en la plataforma Vincúlate permitirá aprovechar plenamente sus capacidades, mejorando significativamente la experiencia del usuario y la accesibilidad a la información. Para asegurar una transición suave, se sugiere realizar esta implementación de manera gradual, comenzando con una fase piloto en una sección específica de la plataforma antes de expandirlo a toda la interfaz.

Es crucial mantener y actualizar regularmente la base de conocimientos del asistente virtual. La expansión continua de esta base, incluyendo la incorporación de nuevos documentos, estudios de caso y otros recursos relevantes, enriquecerá las

capacidades del asistente para responder a un mayor rango de consultas. Además, se recomienda establecer un protocolo de actualización periódica que garantice que el asistente maneje siempre información precisa y actualizada, adaptándose a los cambios en las políticas o procedimientos de Vincúlate.

Para mejorar continuamente la precisión y relevancia de las respuestas del asistente, se sugiere implementar un sistema de retroalimentación de los usuarios. Este sistema permitiría a los usuarios calificar la utilidad de las respuestas recibidas y proporcionar comentarios específicos, lo cual no solo mejoraría la calidad del servicio, sino que también proporcionaría datos valiosos para ajustar y refinar los modelos utilizados.

Dada la naturaleza evolutiva de la inteligencia artificial y las tecnologías de procesamiento de lenguaje natural, es vital mantener una vigilancia tecnológica constante. Esto implica estar al tanto de los avances en modelos de lenguaje, técnicas de Recuperación Aumentada por Generación (RAG) y otras tecnologías relevantes, evaluando periódicamente la posibilidad de incorporar estas mejoras al asistente virtual para mantener su relevancia y eficacia en un entorno tecnológico en rápida evolución.

Se recomienda desarrollar un programa de capacitación para los usuarios de Vincúlate, enfocado en cómo interactuar eficazmente con el asistente virtual. Este programa podría incluir tutoriales en línea, webinars y guías de usuario que expliquen las capacidades y limitaciones del asistente, así como las mejores prácticas para formular consultas. La capacitación ayudará a maximizar el potencial del asistente, asegurando que los usuarios puedan beneficiarse plenamente de sus capacidades.

Considerando el buen desempeño del asistente desarrollado, se sugiere explorar la aplicación de esta tecnología en otras plataformas y bases de conocimiento gestionadas por Concytec. Adaptar el asistente virtual para su uso en repositorios de investigación, sistemas de gestión de proyectos científicos o plataformas de colaboración académica podría mejorar la accesibilidad de la información en estos contextos, demostrando la replicabilidad y escalabilidad del asistente.

Finalmente, se recomienda realizar estudios de seguimiento a largo plazo sobre el impacto del asistente virtual en la productividad y eficiencia de los usuarios de Vincúlate.

Estos estudios podrían proporcionar insights valiosos sobre cómo la inteligencia artificial está transformando el acceso y la utilización de información científica, guiando futuras innovaciones en este campo. Además, se sugiere llevar a cabo investigaciones adicionales sobre el impacto ético y social del uso de IA generativa en plataformas educativas y científicas, asegurando que estas tecnologías se desarrollen y utilicen de manera responsable, maximizando sus beneficios mientras se mitigan los posibles riesgos.

ANEXOS

Anexo 1: Codigo fuente app.py

reocs10 Update app.py f3068ab VERIFIED

raw Copy download link history blame edit delete No virus

```python
import gradio as gr
import pandas as pd
import random
import time

from langchain_openai import ChatOpenAI, OpenAIEmbeddings
from langchain.prompts import ChatPromptTemplate, MessagesPlaceholder
from langchain_community.vectorstores import Chroma
#from langchain.vectorstores import Chroma
from langchain_core.output_parsers import StrOutputParser
from langchain.schema import AIMessage, HumanMessage
from langchain_core.runnables import RunnablePassthrough, RunnableParallel


# Cargar las variables de entorno desde el archivo .env
#load_dotenv()

# Configurar las credenciales de OpenAI
#api_key = os.getenv("OPENAI_API_KEY")
import os

# Obtener la API key de OpenAI de las variables de entorno de Spaces
api_key = os.environ.get("OPENAI_API_KEY")
# Asegúrate de que la API key está configurada
if not api_key:
    raise ValueError("La API key de OpenAI no está configurada en las variables de entorno")

import gradio as gr

class ContextRetriever:
    def __init__(self, embeddings_model, persist_directory, num_results):
        self.persist_directory = persist_directory
        self.num_results = num_results
        self.vectorstore = Chroma(persist_directory=self.persist_directory, embedding_function=embeddings_model)
        self.retriever = self.vectorstore.as_retriever(search_kwargs={'k': self.num_results})

    def get_context(self, question):
        self.context = self.retriever.invoke(question)
        return self.context
```

```python
class ChatHistoryFormatter:
    @staticmethod
    def format_chat_history(chat_history, len_history=10):
        formatted_history = []
        if len(chat_history) >0:
            for human, ai in chat_history[-len_history:]:
                formatted_history.append(HumanMessage(content=human))
                formatted_history.append(AIMessage(content=ai))
            return formatted_history
        else:
            return chat_history

class QuestionContextualizer:
    def __init__(self, chat_model):
        self.chat_model = chat_model
        self.contextualize_q_system_prompt = """Dada una historia de chat y la última pregunta del usuario \
                                              que podría hacer referencia al contexto en el historial del chat, formula una pregunta autónoma \
                                              que pueda entenderse sin el historial del chat. NO respondas la pregunta, \
                                              solo reformúlala si es necesario y, de lo contrario, devuélvela tal como está. \
                                              Esta base de conocimiento está compuesta por contenido de las herramientas de la plataforma Vínculate"""
        self.contextualize_q_prompt = ChatPromptTemplate.from_messages([
            ("system", self.contextualize_q_system_prompt),
            MessagesPlaceholder("chat_history", optional=True),
            ("human", "{question}")
        ])
        self.contextualize_chain = self.contextualize_q_prompt | chat_model | StrOutputParser()

    def contextualize_question(self, formatted_chat_history, question):
        self.question = question
        self.formatted_chat_history = formatted_chat_history
        self.contextualized_question = self.contextualize_chain.invoke({'chat_history': formatted_chat_history, 'question': question})
        return self.contextualized_question
```

```python
class QuestionContextualizer:
    def __init__(self, chat_model):
        self.chat_model = chat_model
        self.contextualize_q_system_prompt = """Dada una historia de chat y la última pregunta del usuario \
                                              que podría hacer referencia al contexto en el historial del chat, formula una pregunta autónoma \
                                              que pueda entenderse sin el historial del chat. NO respondas la pregunta, \
                                              solo reformúlala si es necesario y, de lo contrario, devuélvela tal como está. \
                                              Esta base de conocimiento está compuesta por contenido de las herramientas de la plataforma Vínculate"""
        self.contextualize_q_prompt = ChatPromptTemplate.from_messages([
            ("system", self.contextualize_q_system_prompt),
            MessagesPlaceholder("chat_history", optional=True),
            ("human", "{question}")
        ])
        self.contextualize_chain = self.contextualize_q_prompt | chat_model | StrOutputParser()

    def contextualize_question(self, formatted_chat_history, question):
        self.question = question
        self.formatted_chat_history = formatted_chat_history
        self.contextualized_question = self.contextualize_chain.invoke({'chat_history': formatted_chat_history, 'question': question})
        return self.contextualized_question

class AnswerGenerator:
    def __init__(self, chat_model):
        self.chat_model = chat_model
        # Create the prompt template
        self.prompt_template = """Eres un asistente para tareas de respuesta a preguntas del contenido de las herramientas de la Plataforma Vínculate,
                                  tus usuarios serán investigadores, tecnólogos, innovadores y gestores interesados en los servicios y herramientas de la plataforma
                                  además apoyaras en identificar los formatos propuestos para formalizar acuerdos, convenios y/o contratos en materia de investigación,
                                  desarrollo tecnológico e innovación para su uso en la vinculación Academia-Industria en el Perú.
                                  Los siguientes son fragmentos del contenido de las herramientas en la plataforma.
                                  Brinda la mayor cantidad de información al ususario para que tenga claros los conceptos que está consultando.
                                  Si no sabe la respuesta, simplemente diga que no la sabe.Haga preguntas de seguimiento si corresponde.


                                  Contexto: {context}
                                  Pregunta: {contextualized_question}
                                  Respuesta:
                                  """
        self.prompt_template = ChatPromptTemplate.from_template(self.prompt_template)
        # Question and answer chain that takes in the context and contextualized question and spits out the answer
        self.rag_chain = self.prompt_template | self.chat_model | StrOutputParser()
```

```python
class ResultFormatter:
    @staticmethod
    def format_result(answer, context):
        # Extract unique pairs of titles and video IDs from the context
        unique_documents = set((doc.metadata['titulo'], doc.metadata['url']) for doc in context)
        # Create a plain text string where each title is followed by its URL
        titles_with_links = titles_with_links = [f"{titulo}:{url}" for titulo, url in unique_documents]
        # Join these entries with line breaks to form a clear list
        titles_string = '\n'.join(titles_with_links)
        titles_formatted = f"Links Relevantes:\n{titles_string}"
        # Combine the answer from the result with the formatted list of video links
        response = f"{answer}\n\n{titles_formatted}"

        return response


class VinculateBot:
    def __init__(self, temp=0.9,
                       chat_model_name='gpt-3.5-turbo',
                       embeddings_model_name='text-embedding-3-small',
                       num_results=10,
                       persist_directory='embedding_data',
                       ):

        self.embeddings_model_name = embeddings_model_name
        self.temp = temp
        self.chat_model_name = chat_model_name
        self.num_results = num_results
        self.persist_directory = persist_directory

        # Embeddings Model
        self.embeddings_model = OpenAIEmbeddings(api_key=api_key, model=self.embeddings_model_name)

        # Initialize Chat Model
        self.chat_model = ChatOpenAI(api_key=api_key, temperature=self.temp, model=self.chat_model_name)

        # Set up vector store and other components
        self.context_retriever = ContextRetriever(embeddings_model=self.embeddings_model, persist_directory=self.persist_directory, num_results=self.num_results)
        self.chat_history_formatter = ChatHistoryFormatter()
        self.question_contextualizer = QuestionContextualizer(self.chat_model)
        self.answer_generator = AnswerGenerator(self.chat_model)
        self.result_formatter = ResultFormatter()
```

```python
    def process_chat(self, new_question, chat_history):
        formatted_chat_history = self.chat_history_formatter.format_chat_history(chat_history)
        contextualized_question = self.question_contextualizer.contextualize_question(formatted_chat_history, new_question)
        context = self.context_retriever.get_context(contextualized_question)
        answer = self.answer_generator.generate_answer(context=context, question=contextualized_question)
        final_result = self.result_formatter.format_result(answer, context)
        return final_result


# Datos de ejemplo
users = {"user1": "password1", "user2": "password2"}


api_key = os.environ.get("OPENAI_API_KEY")

# Leer las credenciales de los usuarios desde las variables de entorno
users = {
    os.getenv("USER1"): os.getenv("PASS1"),
    os.getenv("USER2"): os.getenv("PASS2"),
    os.getenv("USER3"): os.getenv("PASS3")
}

# Importar los datos y eliminar la columna "URL"
url = "https://www.dropbox.com/scl/fi/r4drbikkcumcnwsao0qvx/database_vinculate.xlsx?rlkey=zazil8gk89rt0gz8u680i7bqr&st=0r1ougz6&dl=1"

metadata = pd.read_excel(url, engine='openpyxl')
if 'nombre_archivo' in metadata.columns:
    metadata = metadata.drop(columns=['nombre_archivo'])

vinculate_bot = VinculateBot()


import logging
logging.basicConfig(level=logging.DEBUG)
```

```python

def authenticate(username, password):
    user_authenticated = users.get(username) == password
    print(f"Authentication attempt for {username}: {'Success' if user_authenticated else 'Failure'}")
    return user_authenticated

#def logout():
#    return gr.update(visible=False)

def respond(message, chat_history):
    response = vinculate_bot.process_chat(new_question=message, chat_history=chat_history)
    chat_history.append((message, response))
    return "", chat_history

def filter_data(query):
    filtered_data = metadata[metadata.apply(lambda row: query.lower() in row.astype(str).str.lower().to_list(), axis=1)]
    return filtered_data

with gr.Blocks(theme=gr.themes.Soft()) as demo:
    gr.Markdown("""<h1 style="text-align: center;">Asistente Virtual para Herramientas de la Plataforma Vincúlate</h1>""")
    gr.Markdown("""Este asistente virtual utiliza Recuperación Aumentada por Generación (RAG) y el modelo de lenguaje GPT-3.5 Turbo para responder preguntas sobre las her
    with gr.Row():
        with gr.Column(scale=1):
            gr.Markdown("## Base de documentos")
            table = gr.DataFrame(value=metadata)

        with gr.Column(scale=4):
            gr.Markdown("## Chatbot")
            chatbot = gr.Chatbot(height=450)
            msg = gr.Textbox(placeholder="Realiza una pregunta sobre las herramientas de Vinculate", show_label=False)
            submit_btn = gr.Button("Consultar", variant="primary")
            msg.submit(respond, [msg, chatbot], [msg, chatbot])
            submit_btn.click(respond, [msg, chatbot], [msg, chatbot])
            clear = gr.ClearButton([msg, chatbot])

            gr.Markdown("### Ejemplos")
            examples = gr.Examples(examples=["Que es la guía Tanguis", "Define los niveles de madurez tecnologico TRL", "Ejemplos de paquetes tecnológicos"], inputs=msg)

            #logout_button = gr.Button(value="Logout")
            #logout_button.click(logout, [], [], queue=False)


demo.launch(auth=authenticate)
```

Anexo 2: Código fuente requeriments.txt

main / Vinculate_Bot / requirements.txt

reocs10 Update requirements.txt 4e5843b VERIFIED

raw | Copy download link | history | blame | edit | delete | No virus

```
chromadb==0.5.3
geopandas==0.13.2
gradio==4.37.2
gradio_client==1.0.2
langchain==0.2.6
langchain-chroma==0.1.2
langchain-community==0.2.6
langchain-core==0.2.11
langchain-openai==0.1.14
langchain-text-splitters==0.2.2
openai==1.35.10
openpyxl==3.1.5
pandas==2.0.3
pandas-datareader==0.10.0
pandas-gbq==0.19.2
pandas-stubs==2.0.3.230814
python-dotenv==1.0.1
sklearn-pandas==2.2.0
```

BIBLIOGRAFÍA

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM, 9*(1), 36-45. https://doi.org/10.1145/365153.365168

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Schlaefer, N., & Welty, C. (2010). Building Watson: An overview of the DeepQA project. *AI Magazine, 31*(3), 59-79. https://doi.org/10.1609/aimag.v31i3.2303

Al-Amin, M., Ali, M. S., Salam, A., Khan, A., Ali, A., Ullah, A., Alam, M. N., & Chowdhury, S. K. (2024). History of generative artificial intelligence (AI) chatbots: Past, present, and future development. arXiv. https://doi.org/10.48550/arXiv.2402.05122

Park, J. (2024). Development of dental consultation chatbot using retrieval augmented LLM. *The Journal of The Institute of Internet, Broadcasting and Communication, 24*(2), 87–92. https://doi.org/10.7236/JIIBC.2024.24.2.87

Jeong, C. (2023). A study on the implementation of generative AI services using an enterprise data-based LLM application architecture. *Advances in Artificial Intelligence and Machine Learning, 3*(4), 1588-1618. http://dx.doi.org/10.54364/AAIML.2023.1191

Sharma, S., Yoon, D. S., Dernoncourt, F., Sultania, D., Bagga, K., Zhang, M., Bui, T., & Kotte, V. (2024). Retrieval augmented generation for domain-specific question answering. arXiv. https://doi.org/10.48550/arXiv.2404.14760

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv. https://doi.org/10.48550/arXiv.2005.14165

Brynjolfsson, E., Li, D., & Raymond, L. (2023). Generative AI at work. arXiv. https://doi.org/10.48550/arXiv.2304.11771

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gómez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://doi.org/10.48550/arXiv.1706.03762

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). Large language models: A survey. arXiv. https://doi.org/10.48550/arXiv.2402.06196

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv. https://doi.org/10.48550/arXiv.2005.11401

Caelen, O., & Blete, M.-A. (2023). *Developing apps with GPT-4 and ChatGPT.* O'Reilly Media, Inc.

Kapsalis, A. (2024). *Chatbot for EU legislation* [Tesis de maestría, International Hellenic University]. Repositorio Institucional IHU. https://repository.ihu.edu.gr//xmlui/handle/11544/30419

Borek, C. (2024). Comparative evaluation of LLM-based approaches to chatbot creation: Implementing a death doula chatbot [Tesis de maestría, Tampere University]. Repositorio Trepo. https://trepo.tuni.fi/bitstream/handle/10024/154995/BorekCecylia.pdf

Bink, J. M. (2023). *Personalized response with generative AI: Improving customer interaction with zero-shot learning LLM chatbots* [Tesis de maestría, Eindhoven University of Technology]. Research Portal TU/e. https://research.tue.nl/en/studentTheses/personalized-response-with-generative-ai

Balakrishnan, G., & Purwar, A. (2024). Evaluating the efficacy of open-source LLMs in enterprise-specific RAG systems: A comparative study of performance and scalability. arXiv. https://doi.org/10.48550/arXiv.2406.11424

Sánchez-Rivero, R., Ortiz-Reyes, J., Alfonso, J., Romero-González, B., & Simón-Cuevas, A. (2024, March). I-AVATAR: Asistente virtual basado en IA para un soporte

efectivo. *Informática 2024 Conference*. La Habana, Cuba. https://www.researchgate.net/publication/379237517_I-AVATAR_Asistente_Virtual_Basado_en_IA_para_un_Soporte_Efectivo

Guest, G., Bunce, A., & Johnson, L. (2006). How many interviews are enough? An experiment with data saturation and variability. *Field Methods*, 18(1), 59-82. https://doi.org/10.1177/1525822X05279903

Fernandini, C. (2020). *Guía metodológica Tangüis: Propuesta de convenios y contratos en materia de investigación, desarrollo tecnológico e innovación para su uso en la vinculación Academia - Industria en el Perú*. Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica - Concytec. https://hdl.handle.net/20.500.12390/2189

Amazon Web Services. (2023). ¿Qué es la generación aumentada por recuperación? https://aws.amazon.com/es/what-is/retrieval-augmented-generation/

NVIDIA. (2023a). Generative AI chatbot workflow technical brief. https://resources.nvidia.com/en-us/generative-ai-chatbot-workflow/knowledge-base-chatbot-technical-brief?ncid=no-ncid

NVIDIA. (2023b). Generative AI glossary. https://www.nvidia.com/en-us/glossary/generative-ai/

NVIDIA. (2023c). Large language models glossary. https://www.nvidia.com/en-us/glossary/large-language-models/

NVIDIA. (2023d). Retrieval augmented generation glossary. https://www.nvidia.com/en-us/glossary/retrieval-augmented-generation/

OpenAI. (2023). OpenAI API documentation. https://platform.openai.com/docs/introduction

Langchain. (2024). Langchain documentation. https://python.langchain.com/docs/get_started/introduction

Chroma. (2024). Chroma documentation. https://docs.trychroma.com/

Gradio. (2024). Gradio documentation. https://www.gradio.app/docs/

Hugging Face. (2024). Hugging Face documentation. https://huggingface.co/docs

Vincúlate. (2024). Acerca de Vincúlate. https://vinculate.concytec.gob.pe/acerca-de-vinculate/