UIDE
Powered by
Arizona State University®

eig

Maestría en

CIENCIA DE DATOS Y MÁQUINAS DE APRENDIZAJE CON MENCIÓN EN INTELIGENCIA ARTIFICIAL

Trabajo previo a la obtención de título de Magister en Ciencia De Datos y Máquinas de Aprendizaje con Mención en Inteligencia Artificial

AUTORES:

GUALOTUÑA GUANO BRYAN DENIS

LLUMIQUINGA PACHACAMA KARLA LIZBETH

POZO PERALTA CRISTINA ESTEFANIA

QUIHUIRI SIMBAÑA JHONATAN PABLO

TUTORES:

Iván Reyes Chacón

Alejandro Cortés López

TEMA

Generación automática de resúmenes científicos

con modelos NLP

Quito – Ecuador

julio – 2025

REINVENTEMOS EL FUTURO

CERTIFICACIÓN DE AUTORÍA

Nosotros, **Bryan Denis Gualotuña Guano, Karla Lizbeth Llumiquinga Pachacama, Cristina Estefanía Pozo Peralta, Jhonatan Pablo Quihuiri Simbaña,** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

Firma del graduado

Bryan Denis Gualotuña Guano

Firma del graduado

Karla Lizbeth Llumiquinga Pachacama

Firma del graduado

Cristina Estefanía Pozo Peralta

Firma del graduado

Jhonatan Pablo Quihuiri Simbaña

AUTORIZACIÓN DE DERECHOS DE PROPIEDAD INTELECTUAL

Nosotros, **BRYAN DENIS GUALOTUÑA GUANO, KARLA LIZBETH LLUMIQUINGA PACHACAMA, CRISTINA ESTEFANÍA POZO PERALTA, JHONATHAN PABLO QUIHUIRI SIMBAÑA**, en calidad de autores del trabajo de investigación titulado ***Generación automática de resúmenes científicos con modelos NLP,*** autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, julio 2025

Firma del graduado

Bryan Denis Gualotuña Guano

Firma del graduado

Karla Lizbeth Llumiquinga Pachacama

Firma del graduado

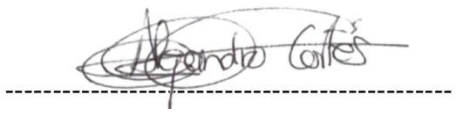
Cristina Estefanía Pozo Peralta

Firma del graduado

Jhonatan Pablo Quihuiri Simbaña

APROBACIÓN DE DIRECCIÓN Y COORDINACIÓN DEL PROGRAMA

Nosotros, **Alejandro Cortés e Iván Reyes** declaramos que los graduados: **BRYAN DENIS GUALOTUÑA GUANO, KARLA LIZBETH LLUMIQUINGA PACHACAMA, CRISTINA ESTEFANÍA POZO PERALTA, JHONATAN PABLO QUIHUIRI SIMBAÑA** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés

Director de la Maestría en
Ciencia de datos y Máquinas
de Aprendizaje mención
Inteligencia Artificial EIG

Iván Reyes

Coordinador de la Maestría
en Ciencia de datos y Máquinas
de Aprendizaje mención
Inteligencia Artificial

DEDICATORIA

A nuestra familia y amigos...

AGRADECIMIENTOS

Agradecemos profundamente el acompañamiento, la guía académica y el apoyo incondicional brindado por nuestros docentes, familiares y amistades, quienes hicieron posible la realización de este proyecto de titulación de maestría.

RESUMEN

El objetivo del presente proyecto es desarrollar una herramienta basada en inteligencia artificial que permita generar automáticamente resúmenes de artículos científicos y facilite la recuperación de información clave. Se consideró la comparativa de cuatro modelos de lenguaje natural según su rendimiento en la tarea de resumen para tener una aplicación integral que aborda la complejidad en textos PDF o documentos escaneados, y que permita realizar resúmenes e interacción con el contenido.

El proyecto está dentro del marco de la investigación aplicada con enfoque tecnológico, empleando un diseño experimental. La evaluación fue desde la perspectiva funcional de la herramienta multifuncional. Se utilizó la técnica de recolección de datos mediante encuesta, aplicada con formularios en Google Forms. Se emplearon criterios de evaluación cuantitativos para medir el rendimiento de los modelos NLP seleccionados.

Se concluye que el modelo integrado en la aplicación **Llama 3.3 y Kimi VL** ofrecen el mejor desempeño para esta tarea en la versión gratuita, generando resúmenes técnicos adecuados para contextos investigativos. Además, permite realizar consultas detalladas sobre el contenido, facilitando el descarte o selección de artículos relevantes, lo cual impacta positivamente en la eficiencia del proceso investigativo en etapa inicial.

Palabras clave

Inteligencia Artificial, Inteligencia Artificial Generativa, Procesamiento del Lenguaje Natural, Deep Learning, Resumen

ABSTRACT

The objective of this project is to develop a program based on artificial intelligence to automatically generate summaries of scientific articles and facilitate the retrieval of key information. The comparison of four natural language models according to their performance in the summarization task was considered to have a holistic application that addresses the complexity in PDF texts or scanned documents and enables summaries and interaction with the content.

The project is within the framework of applied research with a technological approach, with an experimental design. The evaluation was based on the functional perspective of the multifunctional application. A survey data collection technique was used, applied with forms in Google Forms. Quantitative evaluation criteria were used to measure the performance of the selected NLP models.

It is concluded that the model integrated in the **Llama 3.3 y Kimi VL** application offers the best performance for this task in the free version, generating technical summaries suitable for research contexts. In addition, it allows detailed queries on the content, facilitating the discard or selection of relevant articles, which positively impacts the efficiency of the research process in the initial stage.

Key words

Artificial Intelligence, Generative Artificial Intelligence, Natural Language Processing, Deep Learning, Summary

TABLA DE CONTENIDOS

LISTA DE TABLAS

LISTA DE FIGURAS

CAPÍTULO 1

1. INTRODUCCIÓN

1.1. Introducción

En las últimas décadas, el avance tecnológico ha transformado radicalmente el acceso, análisis y gestión de la información científica. No obstante, el crecimiento exponencial de las publicaciones ha generado una sobrecarga informativa que dificulta la identificación de estudios relevantes y la extracción de conocimientos clave a los investigadores. La revisión de documentos extensos demanda un esfuerzo considerable, ralentizando el desarrollo de nuevas investigaciones y afectando la eficiencia en la generación de conocimiento. De ello se puede ver trabajos investigativos en los que se analiza cómo la sobrecarga de información afecta la extracción de puntos clave y se ralentice el análisis profundo sobre una temática en específico (Gordon, 2019). Además, la información esencial suele estar dispersa a lo largo de los textos, lo que complica aún más su análisis y aprovechamiento en el ámbito científico.

La inteligencia artificial ha demostrado ser una herramienta clave en la automatización del procesamiento de textos y en la investigación en diversos países (Díaz Subieta, 2024). Según Díaz Subieta (2024) el uso de herramientas de IA ha generado un incremento de 12.9% anual de investigaciones, en particular menciona que “la investigación académica se hace mucho más eficiente y rápida a través de la IA.” (Díaz Subieta, 2024). En particular, el Procesamiento de Lenguaje Natural (NLP) y el Reconocimiento Óptico de Caracteres (OCR) permiten extraer, interpretar y sintetizar información de manera eficiente, facilitando la comprensión de documentos científicos y optimizando la recuperación de datos relevantes. Por ejemplo,

proponen un sistema automatizado de extracción de datos en tiempo real combinando OCR y NLP para procesamiento de documentos complejos en PDF. (Ganesan & Devi, 2025)

Otros trabajos que abordan el uso de OCR para la extracción e interpretación de información es un enfoque híbrido integrando OCR con modelos de lenguaje LLMs para extraer información y resolver problemas o ambigüedades de documentos escaneados o digitales (Sinha & B., 2024). Debido a la relevancia del tema, este trabajo propone el desarrollo de una herramienta basada en Inteligencia Artificial que transforme la manera en que los investigadores acceden a la literatura científica. Se utilizará una estrategia híbrida para tomar las ventajas de las herramientas OCR, NLP para generar una solución que abarque varias aristas y le permita al investigador tener una aplicación para realizar su trabajo de manera óptima.

El sistema estará compuesto por tres módulos principales: un extractor de texto que procesará tanto documentos digitales como escaneados, se utiliza un modelo OCR para documentos escaneados con alta complejidad. Luego un generador de resúmenes abstractivos basado en modelos avanzados de NLP, en particular se realiza una comparación entre modelos para determinar el óptimo. Finalmente, un ChatBot especializado en la recuperación de información clave dentro de cada artículo, para que el usuario pueda realizar una interacción más enriquecedora con el artículo previamente resumido. Con esta solución, se busca reducir el tiempo de revisión de literatura, optimizar el análisis de datos científicos y mejorar la eficiencia en la generación de conocimiento.

1.2. Definición del proyecto

El proyecto consiste en el desarrollo de una herramienta basada en Inteligencia Artificial que permita generar resúmenes automáticos de artículos científicos mediante técnicas de

Procesamiento de Lenguaje Natural (NLP) y Reconocimiento Óptico de Caracteres (OCR). Esta herramienta incluirá tres módulos principales: un extractor de texto para documentos digitales y escaneados, un generador de resúmenes abstractivos y un ChatBot interactivo para consultas específicas sobre el contenido del artículo.

1.3. Justificación e importancia del trabajo de investigación

El crecimiento exponencial de la literatura científica ha generado una sobrecarga informativa que dificulta la identificación eficiente de estudios relevantes. Investigadores en centros académicos y científicos de la región dedican semanas o incluso meses a la revisión bibliográfica previa al inicio de sus estudios empíricos, lo que ralentiza la producción de nuevo conocimiento. Aunque existen herramientas digitales para la recuperación de información, muchas presentan limitaciones importantes, como baja precisión semántica, incapacidad para interpretar documentos escaneados mediante OCR, y falta de interactividad contextual en entornos especializados.

Se publican aproximadamente 2.5 millones de artículos científicos cada año, lo cual excede la capacidad humana de lectura y análisis, y plantea desafíos técnicos para la extracción automatizada de información desde textos no estructurados (Hong, Ward, Chard, Blaiszik, & Foster, 2021). En consecuencia, se requiere el desarrollo de modelos avanzados que integren técnicas de Procesamiento de Lenguaje Natural (NLP) y Reconocimiento Óptico de Caracteres (OCR), capaces de interpretar documentos complejos y sintetizar contenido relevante de forma automática.

Este proyecto responde directamente a esa necesidad, proponiendo un aplicativo que reduce el tiempo de análisis documental previo a la investigación, y mejora la eficiencia en la

generación de conocimiento científico puesto que se puede interactuar con el articulo y profundizar sobre puntos clave de este. Además, aporta al acceso equitativo a la información académica: los resúmenes generados permiten superar barreras de comprensión asociadas al dominio técnico del idioma, inclusive de idiomas distintos al materno del investigador, facilitando que estudiantes, profesionales de distintas disciplinas accedan a contenido especializado con mayor fluidez.

1.4. Alcance

Para poder desarrollar el presente proyecto de titulación es necesario delimitar el alcance de este. Se parte de un procesamiento de textos, específicamente artículos científicos en varios formatos y se finaliza con el desarrollo de interfaz para realizar tres acciones (cargar documentos, visualizar un resumen e interactuar por medio de un chat bot). Esto implica el desarrollo de tres módulos que abarcan lo siguiente:

1. Procesamiento de información: se procesa artículos científicos en formato PDF, tanto digitales como escaneados o imágenes, utilizando técnicas de extracción y transformación de textos, para documentos escaneados o imágenes se utiliza un modelo OCR.

2. Generación de resúmenes de artículos científicos: se genera resúmenes abstractivos multilingües mediante modelos NLP como: Mistral NEMU, LLaMA 3.3, Kimi VL y MT5 – Small (fine tuning parcial)

3. Desarrollo de una interfaz para el usuario del modelo NLP: se desarrolla una interfaz intuitiva que permite cargar documentos, visualizar resúmenes e interactuar con un

ChatBot. El ChatBot de la interfaz permite realizar consultas específicas sobre el contenido del artículo cargado

1.5. Objetivos

1.5.1. Objetivo general

Desarrollar una herramienta basada en inteligencia artificial que permita generar automáticamente resúmenes de artículos científicos y facilite la recuperación de información clave, con el fin de mejorar la eficiencia de los investigadores en centros científicos de Ecuador para el año 2025.

1.5.2. Objetivo específico

1. Diseñar un módulo de extracción de texto que procese documentos científicos digitales y escaneados mediante técnicas de OCR.
2. Implementar un generador de resúmenes abstractivos utilizando modelos avanzados de NLP adaptados al lenguaje científico.
3. Desarrollar un sistema de preguntas y respuestas tipo ChatBot que permita consultas específicas sobre el contenido del artículo procesado.
4. Crear una interfaz de usuario amigable que integre los módulos de extracción, resumen y consulta interactiva.
5. Evaluar la calidad de los resúmenes generados mediante validación humana y pruebas con usuarios reales.

CAPÍTULO 2

2. REVISIÓN DE LITERATURA

2.1. Marco Teórico

2.1.1. Inteligencia Artificial

La Inteligencia Artificial hace referencia a tareas o actividades que por lo general requieren de la inteligencia humana. Es importante, ya que mejora la eficiencia operativa a través de una toma de decisiones informadas. Comprender los conceptos básicos de la IA, incluyendo el aprendizaje automático, el procesamiento del lenguaje natural y la visión por computadora, es esencial para identificar oportunidades y aplicaciones en diversas áreas. (Mejía Trejo, 2024)

La Inteligencia Artificial es una rama de las ciencias computacionales que se encarga del estudio de modelos de cómputo capaces de realizar actividades propias de los seres humanos con base en dos de sus características primordiales: el razonamiento y la conducta. (López Takeyas, 2007). También, se puede considerar a la Inteligencia Artificial como la capacidad de las máquinas para usar algoritmos simples, intermedios o complejos, aprender de estos datos y utilizarlos en la toma de decisiones. (Rouhiainen, 2018)

El término IA se utiliza a menudo para describir máquinas que imitan funciones cognitivas humanas como el aprendizaje, la comprensión, el razonamiento o la resolución de problemas. (Russell & Norvig, 2016).

Existen dos dimensiones principales tal como se describe en la Tabla 1:

Tabla 1

Dimensiones principales de la Inteligencia Artificial

Sistemas	Concepto
Sistemas que piensan como humanos	Estos sistemas tratan de emular el pensamiento humano; por ejemplo, las redes neuronales artificiales. La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas y aprendizaje. (Mejía Trejo, 2024)
Sistemas que piensan racionalmente	Es decir, con lógica (idealmente), tratan de imitar el pensamiento racional del ser humano; por ejemplo, los sistemas expertos, (el estudio de los cálculos que hacen posible percibir, razonar y actuar). (Mejía Trejo, 2024)
Sistemas que actúan como humanos	Sistemas que actúan racionalmente "El arte de crear máquinas que realicen funciones que requieren inteligencia cuando las realizan personas". (Kurzweil, 1990)
Sistemas que actúan racionalmente	“La Inteligencia Computacional es el estudio del diseño de agentes inteligentes”. (Poole, Mackworth, & Goebel, 1998) "La IA... se ocupa del comportamiento inteligente de los artefactos". (Nillson, 1998)

Nota. Tomado como referencia de (Russell & Norvig, 2016)

2.1.2. Tipos de aprendizaje de la IA

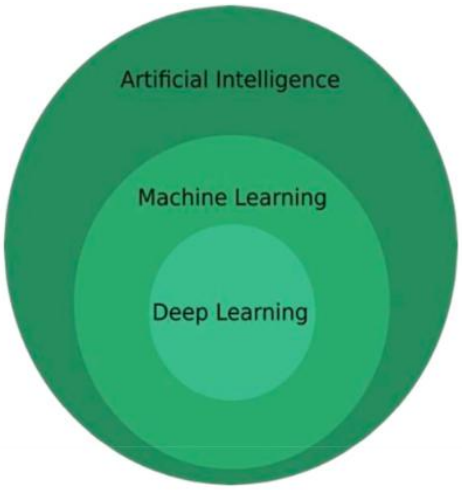
En las décadas de 1990 y 2010, la Inteligencia Artificial experimentó dificultades complejas y brindó soluciones que resultaron útiles y eficaces en diferentes dominios de aplicaciones. (Mejía Trejo, 2024). A medida que evolucionaba el tiempo se desarrollaban herramientas matemáticas más sofisticadas y a la vez intervenían equipos multidisciplinarios lo cual hizo que esta disciplina fuera más rigurosa.

Entre las soluciones principales que se destacan de la Inteligencia Artificial se encuentran: minería de datos, la robótica industrial, la logística, la inteligencia empresarial, el software bancario, el diagnóstico médico, los sistemas de recomendación y los motores de búsqueda. (Mejía Trejo, 2024).

La Inteligencia Artificial es un campo amplio que incluye a dos grandes mundos: Machine Learning y Deep Learning como se explica en la Figura 1.

Figura 1

Machine Learning y Deep Learning



Nota. El grafico identifica los dos campos de la IA.

2.1.3. Machine Learning

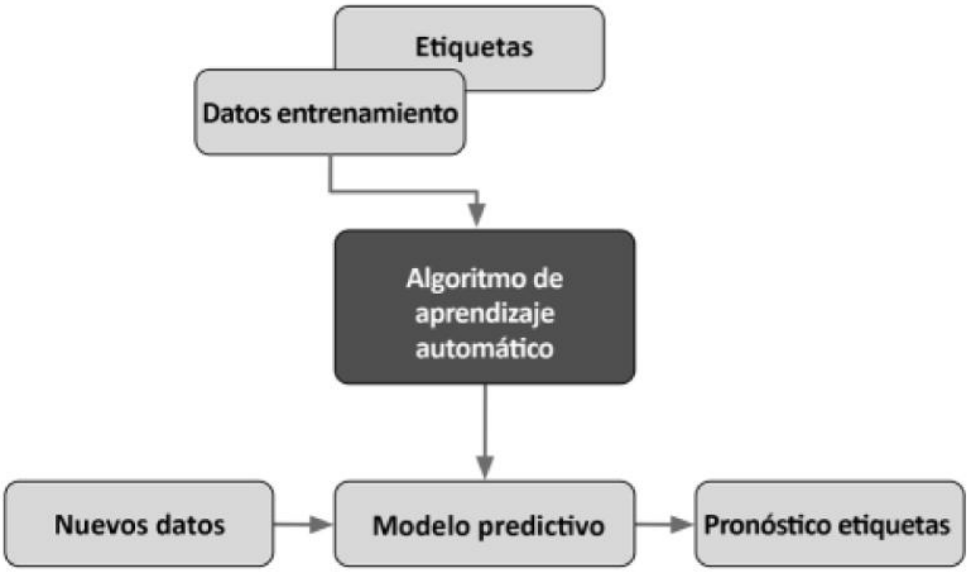
Los 3 tipos de aprendizajes dentro del aprendizaje automático son: Aprendizaje Supervisado, Aprendizaje no Supervisado y el Aprendizaje Reforzado.

Aprendizaje Supervisado

Este tipo de aprendizaje durante la etapa de entrenamiento recibe datos ya conocidos y a partir de estos datos se modela para entender el comportamiento de un usuario o evento y dar una predicción. Es decir, los datos de entrenamiento etiquetados (label) pasan a un algoritmo de aprendizaje automático para que ajuste un modelo predictivo que pueda hacer pronósticos sobre nuevas entradas de datos sin etiquetar. (Raschka, Mirjalili, & Liu, 2023)

Figura 2

Proceso del Aprendizaje Supervisado



Nota. La figura indica el proceso de modelado. (Raschka, Mirjalili, & Liu, 2023)

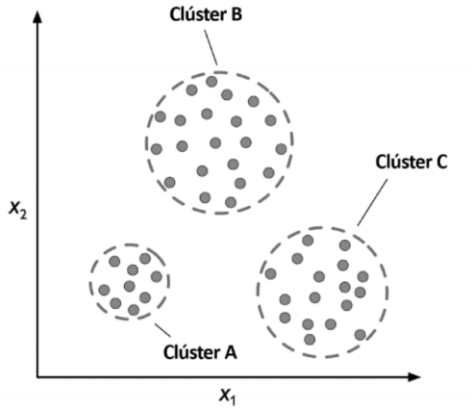
Aprendizaje no Supervisado

En el aprendizaje no supervisado ya no se tiene etiquetas. De acuerdo con los patrones y el comportamiento de los datos se van a crear diferentes grupos también llamados clústeres. Lo principal de este tipo de aprendizaje es que los datos dentro de un mismo clúster comparten características, mientras que, en relación con los otros clústeres tienen diferencias.

El agrupamiento es una técnica de análisis exploratorio de datos o de descubrimiento de patrones que nos permite organizar una gran cantidad de información en subgrupos significativos (clústeres) sin tener ningún conocimiento previo de su pertenencia al grupo. (Raschka, Mirjalili, & Liu, 2023)

Figura 3

Proceso del Aprendizaje no Supervisado



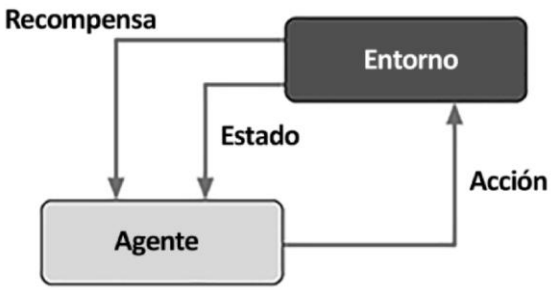
Nota. En la figura se observa que se crearon 3 clústeres. (Raschka, Mirjalili, & Liu, 2023)

Aprendizaje Reforzado

En el aprendizaje reforzado, el objetivo es desarrollar un sistema (agente) que mejore su rendimiento en función de las interacciones con el entorno. (Raschka, Mirjalili, & Liu, 2023)

Figura 4

Proceso del Aprendizaje Reforzado



Nota. En la figura se observa el papel que cumple el Agente maximizando una recompensa. (Raschka, Mirjalili, & Liu, 2023)

2.1.4. Deep Learning

Deep Learning introdujo una arquitectura de red neuronal multicapa que aprende representaciones de datos con niveles de abstracción (LeCun, Bengio, & Hinton, 2015). Dentro de este tipo de redes se encuentran:

Tabla 2

Tipos de Redes

Redes	Networks	Sigla
Redes neuronales profundas	Deep Neural Networks	DNN
Redes neuronales artificiales	Artificial Neural Networks	ANN
Redes neuronales recurrentes	Recurrent Neural Networks	RNN
Redes neuronales convolucionales	Convolutional Neural Networks	CNN

Nota. La tabla resume los 4 principales tipos de redes que existen.

2.1.5. Problemas de la IA

En los últimos años y con el avance de la tecnología, la Inteligencia Artificial ha experimentado un desarrollo a gran escala. Se ha convertido en una herramienta potente en diversas áreas. No obstante, presenta desafíos éticos. Por ejemplo, en el ámbito de la educación: se crea una normativa específica para esta área en respuesta a la necesidad de garantizar que la IA se utilice de manera ética, responsable y equitativa en el ámbito educativo. (Mejía Trejo, 2024).

Para entrenar modelos potentes se requiere de un vasto número de información, lo cual implica técnicas de extracción, procesamiento y transformación. La primera etapa, es decir, la extracción es uno de los temas que generan preocupación especialmente en la privacidad de los usuarios. Las empresas tecnológicas recopilan un gran número de datos de sus usuarios, incluida la actividad en internet, los datos de geolocalización, video y audio. (U.S. Government Accountability Office, 2022).

Uno de los casos más conocidos y publicados fue él de Amazon publicado por CNN Business donde se mencionó que para construir algoritmos de reconocimiento de voz, Amazon, ha grabado millones de conversaciones privadas y han permitido trabajadores las escuchen para transcribirlas algunas de ellas (Valinsky, 2019)

Los desarrolladores de IA argumentan que esta es la única forma de ofrecer aplicaciones valiosas y han desarrollado varias técnicas que intentan preservar la privacidad mientras se obtienen los datos, como la agregación de datos, la desidentificación y la privacidad diferencial. (Russell & Norvig, 2016)

Adicional a los problemas éticos y de privacidad, la presencia de sesgos en los algoritmos también ha sido uno de los motivos de críticas, ya que pueden perpetuar y amplificar la discriminación existente en la sociedad, afectando a grupos marginados y vulnerables. (Valdivia & Tazzioli, 2021). Algunos algoritmos son como cajas negras, es decir no se logra saber en base a que están tomando sus decisiones o que variables pesan más en la decisión, es por eso que la falta de claridad sobre cómo toman decisiones, ha suscitado críticas en torno a la responsabilidad y ética de su uso, especialmente en áreas como la toma de decisiones autónomas. (Pisano, 2022).

El internet de las cosas también ha revolucionado la forma en que se generan los datos por cada segundo, llevando a posible manipulación de la información esto ha generado preocupaciones sobre el control y la manipulación, lo que plantea interrogantes sobre la privacidad y la autonomía individual. (Rubio, 2023). Y, finalmente, los constantes ataques de vulnerabilidad por los ciberdelincuentes presentan un gran desafío constante, ya que su uso puede exponer a las personas a vulnerabilidades y riesgos relacionados con la manipulación de datos sensibles. (Pisano, 2022)

2.1.6. Inteligencia Artificial Generativa

La Inteligencia Artificial Generativa se la puede definir en una sola palabra: Creatividad. Este tipo de tecnologías genera nuevo contenido como imágenes, videos, audios, entre otros, a traces de lenguaje natural conocido como Prompts. En vez de limitarse a conservar las páginas web existentes, Inteligencia Artificial Generativa produce nuevos contenidos. El contenido puede presentarse en formatos que abarcan todas las representaciones simbólicas del pensamiento humano, como: (Mejía Trejo, 2024):

Tabla 3

Contenido en la IA

Tipo de contenido	Ejemplos	Fuente de entrenamiento
Texto	Preguntas a respuestas abiertas, mensajes, resúmenes, historias, poemas.	Libros, blogs, artículos en la web.
Imágenes	Fotografías, dibujos animados	Repositorios de imágenes.
Contenido multimedia	Videos, música.	Repositorio de contenidos.
Código	Fragmentos de código de software	plataformas de desarrollo

Nota. La tabla resume los diferentes contenidos que existen junto con su aplicación.

La Inteligencia Artificial Generativa que produce textos funciona usando un tipo específico de red neuronal conocida como transformador de propósito general. Este transformador se basa en un modelo llamado modelo de lenguaje de gran tamaño, o LLM (por sus siglas en inglés, *Large Language Model*).

Por ello, a los sistemas de generación automática de texto mediante IA se les conoce comúnmente como modelos de lenguaje de gran tamaño.

La Inteligencia Artificial Generativa de textos utiliza un tipo de Red Neuronal Artificial, las RNA son sistemas computacionales que emulan el comportamiento del cerebro humano mediante un conjunto de unidades interconectadas que ajustan sus parámetros para aprender patrones a partir de la información que reciben. (Goodfellow, Bengio, & Courville, 2016)

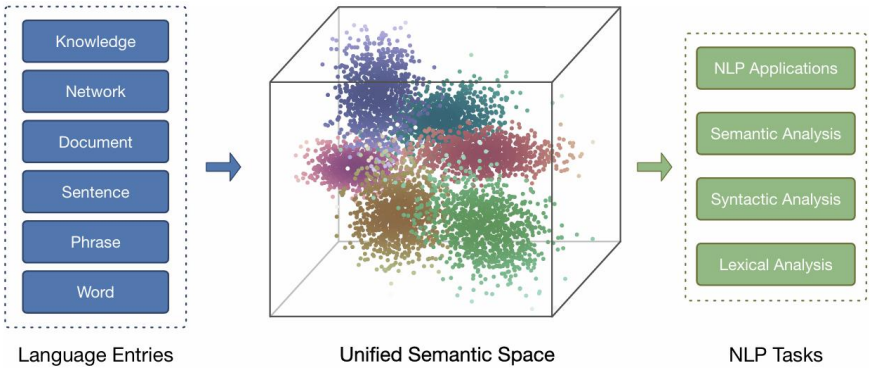
Los sistemas de Inteligencia Artificial Generativa de texto suelen denominarse Large Language Model LLM (modelos de lenguaje de gran tamaño). El tipo de LLM utilizado por la IAGen se conoce como transformador generativo preentrenado o GPT (Generative Pre-trained Transformer) siendo la marca ChatGPT de OpenAI el más conocido. (UNESCO, 2024)

2.1.7. Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural (PLN) se enfoca en desarrollar sistemas capaces de interpretar el lenguaje de los seres humanos a través de representaciones lingüísticas específicas. Esta tarea resulta compleja, difícil y pesada, debido a la naturaleza no estructurada de los textos, distintos niveles de análisis sobre un mismo texto y diversos dominios de origen. Frente a estos retos el Procesamiento de Lenguaje Natural ha demostrado ser una estrategia eficaz para unificar el significado semántico en un espacio común, facilitando así la generalización y el desarrollo de modelos más robustos frente a la variabilidad del lenguaje. (Zhang, Yin, & & Zhang, 2020)

Figura 5

Espacio Semántico en el Procesamiento de Lenguaje Natural

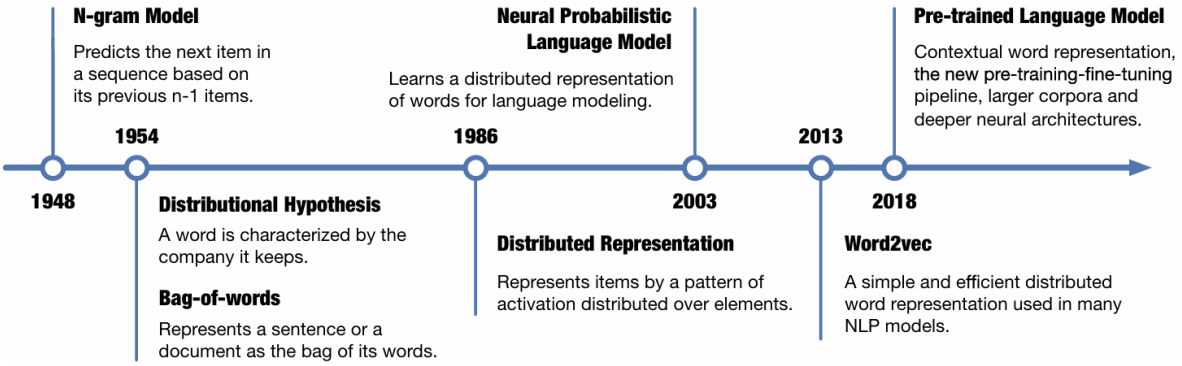


Nota. El uso de representaciones distribuidas permite establecer un espacio semántico unificado que facilita la comprensión de distintos niveles del lenguaje y puede aplicarse a diversas tareas del procesamiento de lenguaje natural. (Zhang, Yin, & & Zhang, 2020)

A lo largo del tiempo, el desarrollo del Procesamiento de Lenguaje Natural ha estado marcado por el crecimiento del poder computacional y la expansión de los datos textuales disponibles. Estos avances han permitido que las representaciones distribuidas, entrenadas con redes neuronales sobre grandes conjuntos de texto, se consoliden como una de las principales estrategias utilizadas por las empresas en sus diferentes áreas. (Zhang, Yin, & & Zhang, 2020).

Figura 6

Development of Representation Learning for NLP



Nota. Se presenta en forma de resumen la evolución que ha tenido el Procesamiento de Lenguaje Natural.

2.1.8. Resumen Automático de Textos

El sistema de Procesamiento de Lenguaje Natural (PLN) es muy utilizado en diferentes ámbitos donde la data input se encuentra en forma de texto. Entre las tareas principales están:

- Traducción automática

- Recuperación de Información
- Clasificación de Textos
- Comprensión de Textos
- Sistemas de Preguntas y Respuestas (Q&A)

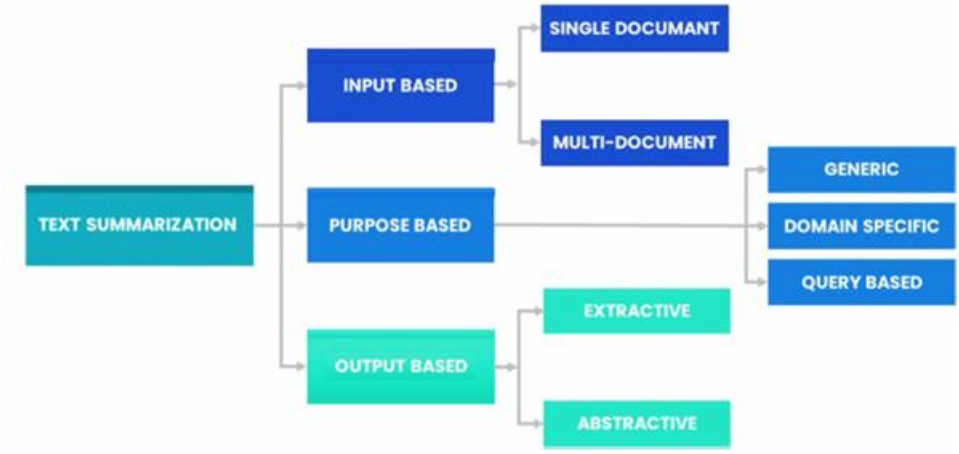
Una de las tareas más relevantes en los últimos años ha sido el resumen automático de textos, el cual se ha convertido en un tema de investigación cada vez más importante. (Kumar & Solanki, 2023). Es un desafío importante dentro de la comprensión del Lenguaje Natural. Se basa en la idea de capturar el significado principal del texto original para producir una representación condensada de esa entrada de texto.

Existen dos sistemas ampliamente utilizados y exitosos cuya diferencia radica en el enfoque. Enfoque extractivo, donde se seleccionan y ensamblan fragmentos del texto para generar una versión abreviada. Y enfoque abstractivo que intenta crear un resumen desde cero, incluyendo aspectos que pueden no aparecer literalmente en el texto original. (Rush, Chopra, & Weston, 2015)

En la era del Big Data, la cantidad de información textual proveniente de diversas fuentes ha crecido de manera exponencial (Kumar & Solanki, 2023). Esta basta cantidad de información contiene valiosos conocimientos y habilidades que deben ser resumidos adecuadamente. Con el aumento en la disponibilidad de documentos, surge la necesidad de desarrollar herramientas de PLN avanzadas para realizar resúmenes automáticos.

Figura 7

Resumen Automático de Textos



Nota. Diagrama de clasificación de la tarea de resumen de texto (text summarization), organizado por tres enfoques: basado en la entrada, en el propósito y en la salida.

2.1.9. Generative Pre-trained Transformer

Generative Pre-trained Transformer Tambien conocido por sus siglas GPT es un transformador generativo pre-entrenado desarrollado por OpenAI que utiliza los Transformers normalmente se lo utiliza para el procesamiento del lenguaje natural. (Natural Language Processing)

La arquitectura de un GPT desde sus inicios en el 2020 ha ido evolucionando, utilizando un corpus más amplio y miles de millones de parámetros. El siguiente cuadro resumen de manera rápida como ha sido su evolución.

Tabla 4

Evolución de los Modelos GPT

Versión	Lanzamiento	Características
GPT-3	jun-20	Primer modelo de gran escala con 175 mil millones de parámetros. (Brown, y otros, 2020)
GPT-3.5	2022	Mejoras en comprensión y coherencia; base de ChatGPT inicial. (OpenAI, 2022)
GPT-4	mar-23	Modelo multimodal, mejora en razonamiento y comprensión. (OpenAI, GPT-4 Technical Report, 2023)
GPT-4o	may-24	Versión mejorada, más rápida y precisa, con capacidades ampliadas. (OpenAI, GPT-4o and more tools for ChatGPT Free, 2024)
GPT-4.1	abr-25	Mejoras en codificación, seguimiento de instrucciones y contexto largo. (OpenAI, GPT-4.1 Release Notes, 2025)
GPT-4.1 mini	may-25	Versión optimizada para tareas rápidas y eficientes. (OpenAI, GPT-4.1 Release Notes, 2025)
GPT-4.1 nano	may-25	Modelo ultraligero para tareas simples con buen rendimiento. (OpenAI, GPT-4.1 Release Notes, 2025)

Nota. En la tabla se visualiza como en la última media década han existido grandes cambios en la forma que se entrenaban los modelos, sus mejoras en codificación y respuesta a tareas.

2.1.10. Ingeniería de Prompts

La ingeniería de Prompts (*prompt engineering*) es el proceso de estructurar entradas para guiar la salida de modelos de inteligencia artificial generativa, como los modelos de lenguaje grande (LLMs) y los generadores de imágenes. Esta práctica ha sido desarrollada tanto en el contexto de generación textual (Ziegler & Berryman, 2023) como visual. (Diab, Herrera, Sleep, Chernow, & & Mao, 2024). Un prompt es un texto en lenguaje natural que describe la tarea que debe realizar una IAGen (Radford, y otros, 2023)

Un prompt es la entrada que se le ingresa al modelo de IAGen con el fin de establecer un contexto, definir la tarea o guiar la generación de la respuesta. El diseño de este puede influir drásticamente en la calidad y relevancia del resultado, puede ser tan simple como una sola palabra, una pregunta o una oración avanzada. (God of Prompts, 2024)

2.1.11. Componentes de un Prompt

Un Prompt puede contener instrucciones, contexto, entrada de datos, indicador de salida como se indica en la siguiente tabla.

Tabla 5

Componente de un Prompt

Elemento	Descripción
Instrucción	Se formula una tarea concreta que se espera del modelo de lenguaje. Por ejemplo, realizar resumen.
Contexto	Se sitúa la instrucción dentro de un escenario específico. Por ejemplo, eres un experto en resúmenes de textos.

Entrada de datos	Se proporcionan elementos previos. Por ejemplo, el texto que se quiere el resumen, que guía la respuesta generada.
Indicador de salida	Se establece el tipo de respuesta que debe entregar la IA. Por ejemplo, resumen de 50 palabras

Nota. La calidad de la respuesta también depende de la calidad de la pregunta.

2.1.12. Técnicas Prompt texto a texto

Son varias a considerar, se tienen no menos de 30 técnicas (Sahoo, y otros, 2024) tales como las lisadas a continuación:

a) Prompt de cadena de pensamiento [COT.Chain-of-Thought Prompting]

El prompt de cadena de pensamiento (CoT. Chain of Thought) es una forma de guiar a los LLM (Large Language Models) para que expliquen cómo resuelven un problema paso a paso, antes de dar la respuesta final. (McAuliffe, 2022)

b) Prompt de conocimiento generado (Generated Knowledge Prompting)

El prompt de conocimiento generado (Generated Knowledge Prompting) es una forma de hacer que un modelo de lenguaje responda mejor a una consulta, pidiéndole que primero cree información relacionada con la consulta y luego la use para dar la respuesta. (Liu, y otros, 2022)

c) Prompts de menor a mayor (Least-to-Most Prompting)

Los Prompts de menor a mayor (Least-to-Most Prompting) son una forma de hacer que un modelo de lenguaje resuelva un problema por pasos. De esta manera, el modelo se basa en las respuestas de los pasos anteriores para resolver los pasos posteriores. (Zhou, y otros, 2022)

d) Prompts basados en complejidad (Complexity-Based Prompting)

Los Prompts basados en la complejidad (Complexity-Based Prompting) son una forma de hacer que un modelo de lenguaje resuelva un problema por pasos y lo explique, usando diferentes maneras de guiarlo. Después, la forma que elige la respuesta final es la que más coincide con las otras maneras que usó. (Fu, Peng, Sabharwal, Clark, & y Khot, 2022)

e) Prompt de decodificación de autocoherencia (Self-Consistency Decoding)

El prompt de decodificación de autocoherencia (Self-Consistency Decoding) realiza varios avances en la cadena de pensamiento, luego selecciona la conclusión más comúnmente alcanzada entre todos los avances. (Wang, y otros, 2022). Si los avances están en desacuerdo en gran medida, se puede consultar a un humano para obtener la cadena de pensamiento correcta. (Diao, Wang, Lin, & Zhang, 2023)

f) Prompts de árbol de pensamiento (Three of Thought Prompting)

Los Prompts de tipo árbol de pensamiento amplían la lógica de la cadena de pensamiento tradicional, ya que, en lugar de seguir una única línea de razonamiento, le piden al modelo que proponga varias alternativas de pasos siguientes. Luego, estas opciones se examinan utilizando distintos métodos de búsqueda, como la búsqueda en amplitud o la búsqueda en viga, que permiten recorrer y comparar diferentes caminos posibles hacia una solución. (Yao, y otros, 2023)

g) Prompts de incitación mayéutica (Maieutic Prompting)

Son similares al árbol del pensamiento. Se le pide al modelo que responda una pregunta con una explicación. Luego se le solicita al modelo que explique partes de la explicación, y así sucesivamente. (Jung, y otros, 2022).

h) Prompt de estímulo direccional (Directional-Stimulus Prompting)

Los prompt de estímulo direccional (Directional-Stimulus Prompting) incluyen una pista como palabras clave deseadas, para guiar un modelo de lenguaje hacia el resultado deseado. (Li, y otros, 2023)

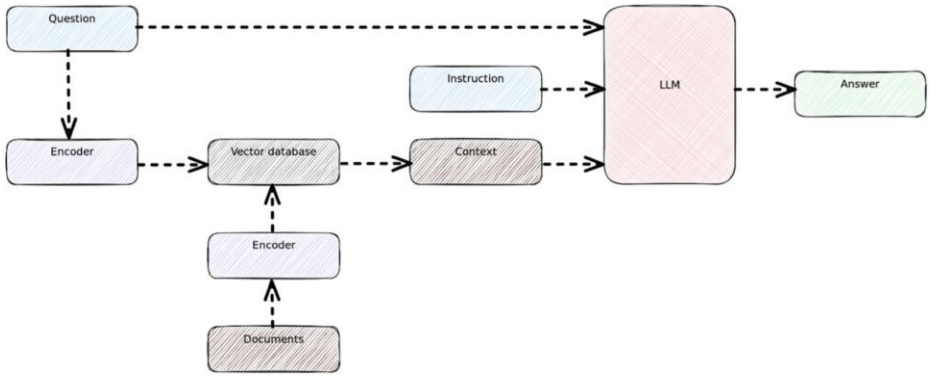
i) Prompt generación recuperación aumentada (RAG. Retrieval Augmented Generation)

Los prompt de generación de recuperación aumentada (Retrieval-Augmented Generation, RAG) son un proceso de dos fases que implica la recuperación de documentos y la formulación de respuestas por parte de un LLM (Large Language Model). (Mejía Trejo, 2024)

En la fase inicial se utilizan incrustaciones densas para recuperar documentos. Esta recuperación puede basarse en una variedad de formatos de bases de datos según el caso de uso, como una base de datos de vectores (vector database), un índice de resumen (summary index), un índice de árbol (tree index) o una tabla de índices de palabras (keyword table index). (Llamaindex, 2024)

Figura 8

Funcionamiento de Retrieval-Augmented Generation



Fuente: Lewis et al. (2020)

Nota. La figura ilustra cómo, a partir de una pregunta del usuario, el sistema utiliza un codificador para buscar información relevante en una base de datos vectorial generada previamente a partir de documentos. Luego, esta información se combina con instrucciones y se envía al modelo de lenguaje (LLM), que finalmente produce una respuesta. (Lewis, y otros, 2020)

Los Prompt RAG son notables por su uso de aprendizaje de pocas instancias, donde el modelo utiliza un pequeño número de ejemplos, a menudo recuperados automáticamente de una base de datos, para informar sus resultados. (Lewis, y otros, 2020)

2.1.13. Tipos de ataques de inyección prompt

Los tipos comunes de ataques de inyección prompt, son:

- **Jailbreaking**. El usuario solicita al modelo interpretar un personaje, responder con argumentos, o finjir ser superior a las instrucciones de moderación. (Xiang, 2023)
- **Fuga de Prompts (Prompt Leaking).** El usuario persuade al modelo para divulgar una pre-indicación que normalmente está oculta para los usuarios. (Xiang, 2023)
- **Contrabando de tokens (Token Smuggling).** El usuario proporciona una tarea o indicación maliciosa está envuelta en una tarea de escritura de código. (Xiang, 2023)

La inyección prompt puede ser vista como un ataque de inyección de código utilizando la ingeniería de Prompts rivales (adversarial prompt engineering). En 2022, el Grupo NCC caracterizó la inyección de Prompts como una nueva clase de vulnerabilidad de sistemas de IAGen/Machine Learning. (Selvi, 2022)

Los ataques de inyección de Prompts son considerados una amenaza, ya que aprovechan debilidades en los sistemas para alterar su funcionamiento. Ante esto, se han propuesto distintas estrategias para mitigar sus efectos. Estas medidas buscan reforzar la seguridad y limitar la posibilidad de manipulación por parte de usuarios malintencionados. Estas incluyen: filtrado de entrada, filtrado de salida, aprendizaje reforzado a partir de comentarios humanos e ingeniería prompt para separar la entrada del usuario de las instrucciones (Perez & Ribeiro, 2022)

2.1.14. Uso de Prompts

a) Prompt nivel intermedio.Un Prompt de nivel intermedio seria la solicitud de una receta de cocina dado los ingredientes que se poseen junto con restricciones o preferencias. Por ejemplo: Dado estos {ingredientes} necesito una receta de {tipo} que no incluya los {ingredientes}. Dame una opción fácil y rápida.

b) Prompt nivel experto: Este tipo de Prompt incluye una experiencia con una comprensión más profunda del tema solicitado. Por ejemplo: Hola Chat GPT estoy investigando sobre el {tema} y tengo problemas en entenderlo. Explícalo de manera equivalente a {referencia o nivel requerido}. Incluye {temas que se necesitan}

c) Prompt nivel maestro: Está diseñado para usuarios con un alto nivel de experiencia en ingeniería de Prompts. Por ejemplo: Imagina que te encuentras en una posición de liderazgo en el año {año futuro} frente a un debate sobre la transformación radical de un sistema clave relacionado con el área de {tema general}. Tu tarea consiste en: a) {Peticion} el estado actual del {tema o sistema}, b) Proponer un plan integral de transformación, considerando factores {económicos, tecnológicos, sociales, políticos,

etc.}, c) Anticipar posibles obstáculos y ofrecer estrategias para enfrentarlos, d) Analizar los impactos de dicha transformación en el contexto {nacional, internacional, ambiental, institucional, etc.

2.1.15. Técnicas utilizadas

1. **Fine Tuning:** El fine tuning o ajuste fino es una técnica del Aprendizaje Automático o Machine Learning que se basa en tomar un modelo previamente entrenado y adaptarlo a una tarea específica utilizando un conjunto de datos más pequeño. Según (Howard & Ruder, 2018), esta técnica permite mejorar el rendimiento en tareas concretas sin necesidad de entrenar un modelo desde cero, lo que ahorra tiempo y recursos computacionales como el uso de GPU's.

2. **Web Scraping**: El web scraping es una técnica automatizada para extraer datos de sitios o páginas web. Esta herramienta resulta útil cuando se requiere recolectar grandes cantidades de información de forma estructurada desde fuentes en línea. De acuerdo con (Mitchell, 2015) el web scraping implica el uso de scripts o bots para navegar por sitios web y capturar contenidos relevantes. En el contexto del presente trabajo, se empleó esta técnica para construir un dataset personalizado tomando como referencias la tesis de la Universidad Internacional del Ecuador.

3. **Google/mt5-small:** Para el entrenamiento del sistema de generación automática de resúmenes, se empleó el modelo Google/mt5-small, variante más ligera desarrollada por Google (huggingface, 2025). El modelo google/mt5-small, utilizado como punto de partida para la generación automática de resúmenes académicos, fue preentrenado únicamente sobre el corpus multilingüe mC4 mediante tareas de reconstrucción de texto (*span corruption*), sin haber

recibido entrenamiento supervisado en tareas específicas como la síntesis textual (Xue, y otros, 2020).

4. **Optimizador AdamW:** El optimizador AdamW o desacoplador de decaimiento de pesos por su nombre en inglés “Adam with decoupled weight decay”, se trata de una variante del algoritmo Adam que mejora la regularización del proceso, puesto que separa el término de decaimiento de pesos de la actualización de gradientes (Loshchilov & Hutter, 2019). Este desacoplamiento del regularizador evita que se interfiera con el mecanismo adaptativo de las tasas de aprendizaje, lo cual da como resultado que se genere una mayor estabilidad en el entrenamiento y una mejor generalización por parte de modelos con arquitecturas profundas como el mt5-small (Loshchilov & Hutter, 2019)

CAPÍTULO 3

3. DESARROLLO

3.1. Desarrollo del Trabajo

Este capítulo explica paso a paso cómo se desarrolló el sistema que permite generar resúmenes automáticos a partir de documentos científicos en formato PDF. Se abordan aspectos como la estructura general del sistema, cómo se procesan distintos tipos de documentos, la incorporación de modelos de inteligencia artificial para crear los resúmenes, y el uso de servicios en la nube para almacenar y ejecutar todo el proceso. Durante la construcción del sistema se buscó mantener una estructura organizada, flexible y fácil de escalar, utilizando herramientas de código abierto y servicios en la nube modernos que aseguran buen rendimiento y disponibilidad.

3.2. Arquitectura

Uno de los aspectos más importantes al momento de construir el sistema fue definir una estructura clara y bien organizada que pudiera crecer fácilmente con el tiempo. Por eso, se decidió separar las funciones del sistema en dos partes principales: el Frontend, que es la interfaz gráfica con la que interactúa el usuario, y el Backend, que se encarga de todo el procesamiento y la lógica interna. Esta forma de organizar el sistema facilita mucho su mantenimiento, permite agregar nuevas funciones sin complicaciones y hace más sencilla la conexión con otros servicios y modelos de inteligencia artificial.

3.2.1. Frontend con Streamlit

El Frontend del sistema se desarrolló usando Streamlit, una herramienta que permite crear aplicaciones web de forma rápida y sencilla con Python. Una de sus principales ventajas es el despliegue fácil y compatible con lenguajes tradicionales, como HTML, CSS o JavaScript, para construir interfaces atractivas e interactivas. Además, Streamlit se integra fácilmente con librerías de procesamiento de texto y modelos de inteligencia artificial, lo que facilita mostrar los resultados de manera inmediata y clara al usuario con lo indica la Figura 9.

En la interfaz, el usuario puede:

- Subir documentos en formato PDF desde su equipo local.

Figura 9

Subir Archivos PDF

```
with col1:
    st.header("📩 Subir PDF")
    uploaded_file = st.file_uploader("Carga tu PDF", type="pdf", accept_multiple_files=False, key=f"upload_{session_key}")

    if st.button("Procesar", key=f"process_{session_key}") and uploaded_file:
        with st.spinner("Procesando..."):
            selected_model = current_session.get("selected_model", "meta/Llama-4-Scout-17B-16E-Instruct")
            result = upload_pdf([uploaded_file], selected_model)
            if result:
                current_session["session_id"] = result.get("session_id")
                summary = result.get("summary")
                current_session["chat_history"] = []  # Reiniciar chat
                current_session["chat_history"].append(("SciBot", summary))  # Mostrar resumen en el chat
                current_session["has_document"] = True
                st.success("¡Documento procesado!")
                st.rerun()
```

Nota. Acciones para subir archivos Artículos en formato PDF

- Seleccionar el modelo de IA con el cual desea generar el resumen.

Figura 10

DropDown para modelo

```python
with col_model:
    # Menú desplegable de selección de modelo
    model_options = {
        "LLama-3.3": "meta-llama/llama-3.3-70b-instruct:free",
        "Mistral": "mistralai/mistral-nemo:free",
        "Kimi VL": "moonshotai/kimi-vl-a3b-thinking:free",
        "MT5 Small": "mt5-small",
    }

    # Obtiene la selección de modelo actual
    current_model_key = "Kimi VL"  # Por defecto
    for key, value in model_options.items():
        if value == current_session.get("selected_model", "moonshotai/kimi-vl-a3b-thinking:free"):
            current_model_key = key
            break
```

Nota. Programación para lista desplegable para elección del modelo para respuestas

- Visualizar el texto extraído del PDF y resumen generado por el modelo

Figura 11

Visualizador del Chat

```python
    chat_container = st.container()
    with chat_container:
        if current_session["chat_history"]:
            for speaker, msg in current_session["chat_history"]:
                if speaker == "Tú":
                    st.markdown(f"**🧑 {speaker}:** {msg}")
                else:
                    st.markdown(f"**🤖 {speaker}:** {msg}")
        elif not current_session["has_document"]:
            st.info("Primero debes subir un documento a la izquierda.")
        else:
            st.info("¡Documento cargado! Puedes hacer preguntas sobre él.")
```

Nota. Visualizar chat generado en la sesión

Streamlit, al estar basado en scripts de Python, facilita la integración directa con los modelos preentrenados y servicios de Backend, permitiendo construir una experiencia fluida y dinámica para el usuario final.

El código fuente del frontend utilizado en esta investigación está disponible como material complementario (Ver Fichero 1: frontend_scibot.zip).

3.2.2. Backend con Django

El Backend del sistema se desarrolló utilizando Django, un framework web en Python que destaca por ser seguro, confiable y muy completo. Django facilita la organización del proyecto y permite manejar de forma ordenada las diferentes partes de la aplicación, como las solicitudes que llegan del usuario, el procesamiento de datos y la generación de respuestas. Además, es ideal para construir APIs que permiten que el sistema se comunique con otros servicios o aplicaciones, siempre cuidando la seguridad y eficiencia en el manejo de la información.

Las principales funciones del Backend incluyen:

- Recepción y validación de archivos PDF enviados desde el Frontend.
- Identificación automática del tipo de PDF (legible o escaneado).
- Llamadas a servicios externos como Google Cloud Vision.
- Procesamiento del texto mediante los modelos de IA integrados.
- Almacenamiento de resultados en la nube.
- Gestión de errores, tiempos de espera y logs.

Figura 12

Validador de errores

```python
def parse_error(text):
    try:
        message_match = re.search(r"Message:\s*(.+)", text)
        max_size_match = re.search(r"Max size:\s*(\d+)", text, re.IGNORECASE)

        message = message_match.group(1).strip() if message_match else ""
        max_size = max_size_match.group(1) if max_size_match else ""

        return {
            "error": f"{message} (Max size: {max_size})"
        }
    except:
        return {
            "error": "Error inesperado intenta de nuevo"
        }
```

Nota. Se crea un error visual cuando se recibe mensaje desde el modelo

Django también ayudó a organizar bien el proyecto, separando las partes que muestran la información al usuario, las que manejan los datos y la lógica que define cómo funciona todo. Gracias a esta estructura, es mucho más fácil hacer cambios o agregar nuevas funciones en el futuro, como permitir que los usuarios se registren, guardar historiales de uso o el soporte para varios idiomas.

3.2.3. Separación de responsabilidades

Separar el Frontend del Backend no fue solo una decisión técnica, sino también una forma práctica de hacer que el sistema sea más flexible. Esta división permite que cada parte funcione por separado, por lo que, si en el futuro se quiere cambiar la interfaz por otra más avanzada, como una hecha con React, no sería necesario modificar el resto del sistema. Además,

esta forma de trabajar facilita que diferentes personas, con distintos perfiles técnicos, puedan colaborar sin complicaciones.

3.3. Procesamiento de Documentos PDF

Uno de los retos más comunes al trabajar con documentos es que no todos vienen en el mismo formato. En este proyecto, se usaron archivos PDF como punto de partida, pero no todos son iguales. Algunos ya contenían texto digital que podía leerse directamente, mientras que otros eran escaneos de documentos físicos, por lo que el texto estaba dentro de imágenes. Por eso, fue necesario usar distintas herramientas según el tipo de archivo, para poder extraer el contenido correctamente en cada caso.

3.3.1. PDFs legibles: extracción con PDF Plumber

Para los archivos PDF que contienen texto digital, se utilizó la biblioteca PDF Plumber, la cual permite extraer texto preservando el formato estructural original del documento, incluyendo tablas, párrafos, encabezados y pies de página como lo indica la Figura 13.

Figura 13

PDF Plumber

```python
@csrf_exempt
def loadPdf(request):
    if request.method == "POST":

        uploaded_pdf = request.FILES.get('pdf')
        if uploaded_pdf:
            try:
                model = request.GET.get('model')
                with pdfplumber.open(uploaded_pdf.file) as pdf:
                    data = ""
                    for page in pdf.pages:
                        text = page.extract_text()
                        if text:
                            data += text + ' '
                    if data == "":
                        data = utils.get_image_data(uploaded_pdf)
            except:
                return JsonResponse(
                    {"error" : "Error al procesar archivo"},
                      status=400)
```

Nota. Se procesa el PDF con PDF plumber para extraer el texto

Se diseñó un módulo de preprocesamiento que analiza cada página del documento, filtra secciones no relevantes (como números de página o metadatos), y consolida el texto extraído en un formato limpio y listo para ser procesado por los modelos de inteligencia artificial.

3.3.2. PDFs escaneados: uso de Google Cloud Vision para OCR

Para aquellos documentos que consisten en imágenes (por ejemplo, escaneos de artículos físicos), se recurrió a la herramienta de reconocimiento óptico de caracteres (OCR) de Google Cloud Vision API. Esta API permite detectar y transcribir texto contenido en imágenes con alta precisión, gracias a modelos de machine learning entrenados en millones de ejemplos. La implementación se encuentra en la Figura 14.

Figura 14

OCR de Google

```python
from google.cloud import vision
from google.cloud import storage

ocr_client = None
storage_client = None

def load_ocr_model():
    global ocr_client, storage_client
    if ocr_client is None:
        credentials_path = "./scibot-backend-b5fd3808c145.json"
        ocr_client = vision.ImageAnnotatorClient.from_service_account_file(credentials_path)
        storage_client = storage.Client.from_service_account_json(credentials_path)
```

Nota. Se carga Google cloud visión como API para el reconocimiento de imágenes y transformar a texto.

El proceso es el siguiente:

1. El sistema detecta automáticamente si el PDF contiene imágenes en lugar de texto digital.
2. El PDF es almacenado en Google Storage.
3. El PDF es enviado a la API de Cloud Vision.
4. El texto reconocido es retornado y almacenado para el siguiente paso.

Figura 15

Bucket de Almacenamiento

```python
def get_image_data(uploaded_pdf):
    gcs_input_uri = load_file_to_bucket(uploaded_pdf)
    prefix_folder = f"{OUTPUT_FOLDER}/{uuid.uuid4()}/"
    gcs_output_uri = f"gs://{BUCKET_NAME}/{prefix_folder}"

    mime_type = "application/pdf"
    batch_size = 2

    feature = vision.Feature(type_=vision.Feature.Type.DOCUMENT_TEXT_DETECTION)
    gcs_source = vision.GcsSource(uri=gcs_input_uri)
    input_config = vision.InputConfig(gcs_source=gcs_source, mime_type=mime_type)

    gcs_destination = vision.GcsDestination(uri=gcs_output_uri)
    output_config = vision.OutputConfig(gcs_destination=gcs_destination, batch_size=batch_size)

    async_request = vision.AsyncAnnotateFileRequest(
        features=[feature], input_config=input_config, output_config=output_config
    )

    operation = ocr_client.async_batch_annotate_files(requests=[async_request])

    print("Esperando a que termine la operación OCR...")
    operation.result(timeout=420)

    output_text = ""
    blob_list = storage_client.list_blobs(BUCKET_NAME, prefix=prefix_folder)
    for blob in blob_list:
        json_string = blob.download_as_bytes().decode("utf-8")
        try:
            response = json.loads(json_string)
            annotation = response["responses"][0]["fullTextAnnotation"]
            output_text += annotation["text"] + " "
        except:
            pass

    return output_text
```

Nota. Se guarda el texto reconocido por el OCR en los buckets para un proceso posterior

Este enfoque garantiza que incluso documentos escaneados se puedan procesar y resumir por los modelos de IA.

3.4. Almacenamiento y lectura de archivos

Para que todo el sistema funcione correctamente desde que el usuario sube un documento hasta que se genera el resumen final es necesario contar con un lugar seguro y accesible donde guardar todos los archivos. Para eso se utilizó Google Cloud Storage, un servicio en la nube que permite almacenar tanto los documentos originales como los textos ya procesados y los resúmenes generados.

Para ello, se implementó Google Cloud Storage (GCS), configurando buckets específicos para cada tipo de archivo como se refleja en la Figura 16:

1. PDFs originales cargados.
2. Imágenes generadas para OCR.
3. Archivos de texto intermedio.
4. Resultados finales de resúmenes.

Figura 16

Resultados en Bucket

```python
def list_files():
    blob_list = storage_client.list_blobs(BUCKET_NAME, prefix=f"{OUTPUT_FOLDER}")
    print(blob_list)
    for blob in blob_list:
        json_string = blob.download_as_bytes().decode("utf-8")
        print(json_string)
        try:
            response = json.loads(json_string)
            annotation = response["responses"][0]["fullTextAnnotation"]
        except:
            pass

def load_file_to_bucket(uploaded_pdf):
    bucket = storage_client.bucket(BUCKET_NAME)
    filename = f"{FILES_FOLDER}/{uuid.uuid4()}_{uploaded_pdf.name}"
    blob = bucket.blob(filename)
    uploaded_pdf.seek(0)
    blob.upload_from_file(uploaded_pdf, content_type=uploaded_pdf.content_type)

    return f"gs://{BUCKET_NAME}/{filename}"
```

Nota. Se devuelve los valores del bucket para mostrarlos en el chat

Este almacenamiento centralizado asegura que los archivos estén siempre disponibles, protegidos y listos para ser utilizados por cualquier parte del sistema. Además, facilita mucho el trabajo en conjunto entre los distintos módulos, ya que todos pueden acceder a la misma información sin complicaciones.

El código fuente del backend y uso de almacenamiento utilizado en esta investigación está disponible como material complementario (Ver Fichero 2: backend_scibot.zip).

3.5. Modelos de Inteligencia Artificial

Una vez que el sistema ha logrado extraer el texto del documento, se pasa a la parte más importante: generar un resumen automático usando modelos de inteligencia artificial. La idea es

que estos modelos lean el contenido, identifiquen las ideas principales y lo resuman de forma clara, sin perder los puntos clave ni la lógica del documento original.

3.5.1. Modelos open source utilizados

Para esta tarea, se integraron diversos modelos de lenguaje preentrenados y de código abierto, descargados de plataformas como GitHub y Hugging Face. Cada modelo presenta particularidades en cuanto a tamaño, arquitectura y rendimiento en tareas de comprensión lectora y generación de texto.

Los modelos utilizados fueron:

- Mistral NEMU se orienta a la eficiencia y velocidad en tareas lingüísticas complejas.
- LLaMA 3.3 destaca por su capacidad para seguir instrucciones de forma precisa y contextual
- Kimi VL combina visión y lenguaje, lo que lo hace ideal para tareas multimodales;
- MT5-small, aunque de menor tamaño, permite ajustes finos y personalizados para tareas específicas gracias al fine-tuning parcial aplicado.

Cada modelo fue adaptado mediante Prompts personalizados y parámetros ajustados a los requerimientos del proyecto (máximo de tokens, temperatura, tipo de resumen, etc.).

3.6.Modelo Google/mt5-small

Para el entrenamiento del sistema de generación automática de resúmenes, se empleó el modelo Google/mt5-small. Esta característica exige su ajuste mediante fine-tuning supervisado para adaptarlo a la tarea de resumen, lo cual se realizó empleando un corpus compuesto por resúmenes de trabajos de grado y pregrado, seleccionados por su estructura discursiva científica, diversidad temática y disponibilidad de recursos académico-científicos de Ecuador.

El corpus utilizado para el entrenamiento de este modelo corresponde a una base compuesta de resúmenes de trabajos de grado y pregrado de la Universidad Internacional del Ecuador (UIDE). Estos textos fueron recolectados mediante técnicas de web scraping, aplicadas a repositorios institucionales públicos, con el fin de automatizar la extracción del contenido presente en los campos “abstract” y/o “resumen”. A cada registro extraído se le incorporó una columna adicional que contiene una versión condensada, producida de forma semi-automática, la cual sintetiza los elementos clave del documento (objetivo, resultados principales y conclusiones). Esta estrategia permitió generar pares entrada-salida breves, compatibles con las limitaciones computacionales del proyecto, tales como el manejo de tokens, la capacidad de memoria disponible y el tiempo de entrenamiento. A continuación, en la Figura 17 se muestra el código usado para la creación de la síntesis generada de forma semi automática.

Figura 17

Código para la generación de Resúmenes

```python
def summarize_excel(file_path: str, nombre: str) -> None:
    try:
        output_dir = "/content/drive/MyDrive/resumenes"
        os.makedirs(output_dir, exist_ok=True)
        output_path = os.path.join(output_dir, "BASE_UIDE_ABSTRACTS_RESUMENES_PARTE1_ES.xlsx")

        summarizer_model = "facebook/bart-large-cnn"
        summarizer_tokenizer = AutoTokenizer.from_pretrained(summarizer_model)
        summarizer = AutoModelForSeq2SeqLM.from_pretrained(summarizer_model)

        translator_model = "Helsinki-NLP/opus-mt-en-es"
        translator_tokenizer = AutoTokenizer.from_pretrained(translator_model)
        translator = AutoModelForSeq2SeqLM.from_pretrained(translator_model)

        df = pd.read_excel(file_path)
        print(f"Archivo '{file_path}' cargado con {len(df)} filas.\n")

        filtered_df = df[df["Persona"] == nombre].copy()
        if filtered_df.empty:
            print(f"No se encontraron entradas para '{nombre}'.")
            return

        short_summaries_es = []

        for index, row in filtered_df.iterrows():
            print(f"Procesando resumen {index + 1}/{len(filtered_df)}")
            try:
                texto = row['Summary']
                inputs = summarizer_tokenizer.encode(texto, return_tensors="pt", max_length=1024, truncation=True)
                summary_ids = summarizer.generate(
                    inputs,
                    max_length=250,
                    min_length=80,
                    num_beams=4,
                    early_stopping=True
                )
                summary_en = summarizer_tokenizer.decode(summary_ids[0], skip_special_tokens=True)
                translation_inputs = translator_tokenizer.encode(summary_en, return_tensors="pt", max_length=512, truncation=True)
                translated_ids = translator.generate(translation_inputs, max_length=512)
                summary_es = translator_tokenizer.decode(translated_ids[0], skip_special_tokens=True)

                print(summary_es)
                short_summaries_es.append(summary_es)
            except Exception as e:
                print(f"Error en fila {index}: {e}")
                short_summaries_es.append("")

            # Guardar cada 5 filas o al final
            if (index + 1) % 5 == 0 or (index + 1) == len(filtered_df):
                filtered_df.loc[:index, "short_summaries"] = short_summaries_es
                filtered_df.to_excel(output_path, index=False)
                print(f"✅ Progreso guardado hasta la fila {index + 1}")

        print(f"\n✔ Resúmenes traducidos guardados en '{output_path}'")

    except FileNotFoundError:
        print(f"Error: No se encontró el archivo '{file_path}'.")
    except KeyError as e:
        print(f"Error: Falta la columna {str(e)}.")
    except Exception as e:
        print(f"Error inesperado: {e}")

if __name__ == "__main__":
    from google.colab import drive
    drive.mount('/content/drive')

    excel_file = "/content/drive/MyDrive/resumenes/BASE UIDE ABSTRACTS Parte 1.xlsx"
    summarize_excel(excel_file, "PERSONA")
```

Nota. Código usado para la creación de la síntesis generada de forma semi automática.

Después de obtener los resúmenes se validaron manualmente para revisar que el resumen corto tenga claridad y coherencia para poder usarlo de insumo para entrenar el modelo. La base resultante cuenta con 1787 registros y 9 columnas, donde se detalla el año de publicación (Year), el título del proyecto (Title), la dirección web donde se puede ver el documento completo (URL), el nombre del autor(a) y/o autores(as) (Autor), las palabras clave (Keys), el abstract del proyecto (Summary), el número de palabras inicial del abstract (Num Palabras), la persona que revisó el resumen corto (Persona) y la columna adicional con un resumen corto adaptado a las necesidades del proyecto (short_summaries).

3.6.1. Limpieza y análisis de la base de datos.

Durante la fase de preparación del corpus, se implementaron procesos de limpieza y filtrado con el objetivo de estandarizar el texto y asegurar la calidad de los datos utilizados para el entrenamiento del modelo. En primer lugar, se aplicaron funciones para detectar y normalizar caracteres no válidos en las columnas clave (short_summaries y Summary), eliminando símbolos fuera del estándar ASCII, comillas tipográficas, viñetas y espacios irregulares. Como se muestra a continuación en la Figura 18.

Figura 18

Código de limpieza

```python
# Columnas a limpiar
columnas_objetivo = ["short_summaries", "Summary"]

# Función para detectar caracteres no válidos
def detectar_caracteres_raros(texto):
    if pd.isna(texto): return ""
    return "".join(sorted(set(re.findall(r"[^\x00-\x7F]", str(texto)))))

# Función para normalizar texto español
def normalizar_texto(texto):
    if pd.isna(texto): return ""
    texto = str(texto).strip()
    texto = re.sub(r"\s+", " ", texto)  # eliminar espacios extra
    texto = re.sub(r"[“”]", '"', texto)  # comillas raras
    texto = re.sub(r"[‘’]", "'", texto)  # comillas simples
    texto = re.sub(r"[•·●▪]", "-", texto)  # bullets
    return texto  # sin eliminar tildes ni ñ

# Aplicar limpieza y detectar caracteres sospechosos
for col in columnas_objetivo:
    df[f"{col}_corruptos"] = df[col].apply(detectar_caracteres_raros)
    df[col] = df[col].apply(normalizar_texto)

# Mostrar resumen de caracteres sospechosos
for col in columnas_objetivo:
    print(f"\n Columna: {col}")
    print(df[f"{col}_corruptos"].value_counts().head(10))  # caracteres extraños más comunes
```

Nota. Se realizó una limpieza de la Base de Datos.

Posteriormente, se utilizó una función de detección de idioma para truncar automáticamente las entradas que incluían contenido en inglés, conservando únicamente las oraciones en español. Esto fue especialmente importante para asegurar la coherencia lingüística del corpus de entrenamiento. En este paso se guardó una nueva versión de la base para tener un seguimiento del procesamiento de este. En la Figura 19 se puede observar la función desarrollada.

Figura 19

Función para eliminar contenido en inglés

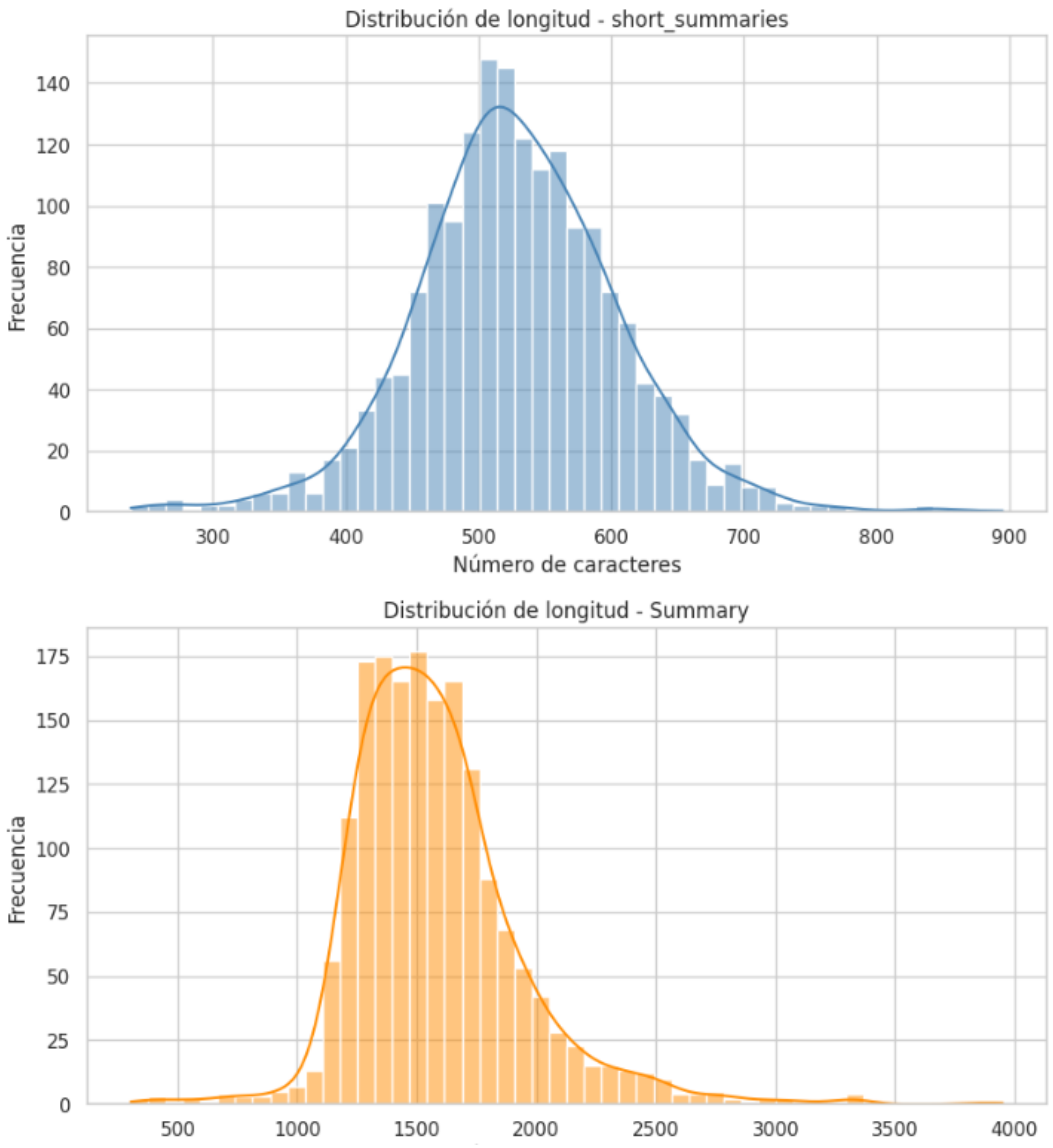
```python
#  Función para eliminar texto en inglés desde la primera oración detectada
def cortar_en_ingles(texto):
    if pd.isnull(texto):
        return texto
    oraciones = re.split(r'(?<=[.!?])\s+', str(texto).strip())
    oraciones_es = []
    for oracion in oraciones:
        try:
            idioma = detect(oracion)
            if idioma == 'es':
                oraciones_es.append(oracion)
            else:
                break
        except:
            oraciones_es.append(oracion)
    return " ".join(oraciones_es).strip()

# Cargar base para eliminar texto en inglés
df = pd.read_excel("/content/drive/MyDrive/modelos_nlp/BASE_UIDE_LIMPIO.xlsx")

# Aplicar la función y guardar en nueva columna
df["Summary"] = df["Summary"].apply(cortar_en_ingles)

# Exportar nueva versión de la base limpia
ruta_salida = "/content/drive/MyDrive/modelos_nlp/BASE_UIDE_LIMPIO.xlsx"
df.to_excel(ruta_salida, index=False)
print("Archivo guardado:", ruta_salida)
```

Nota. Se conservó solo texto en español.

Adicionalmente, se evaluaron las longitudes de los textos, eliminando aquellos registros que no cumplían con el mínimo de caracteres necesarios para representar adecuadamente un resumen (≥100 caracteres en short_summaries y ≥300 en Summary). Este filtrado ayudó a reducir ruido informativo y optimizar la eficiencia computacional durante el entrenamiento. En la limpieza inicial que retiró texto en inglés se obtuvo 23 registros vacíos, por lo que la base se redujo como lo indica la Figura 20.

Figura 20

Histograma de longitud de palabras por resumen y resumen corto



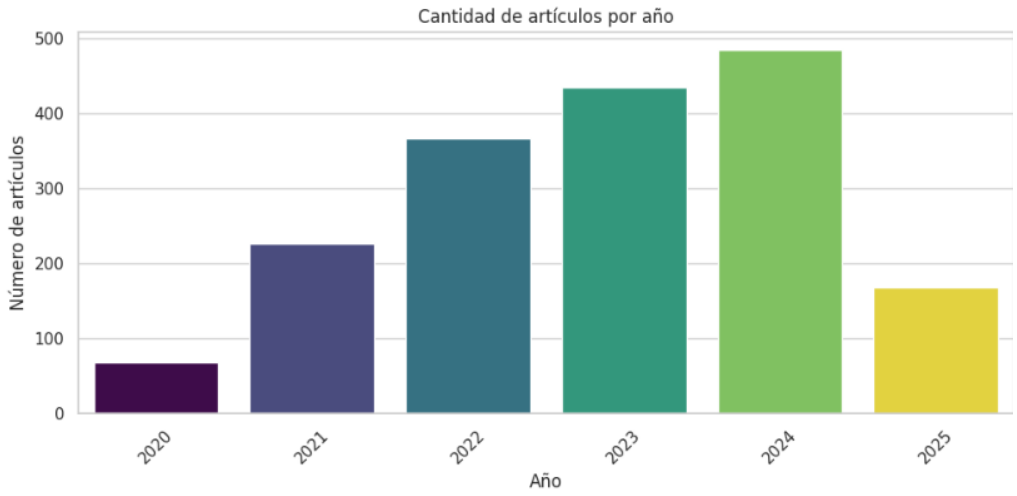
Nota. Se eliminó el ruido informativo para optimizar la eficiencia computacional

Como se puede observar en el gráfico, la cantidad de caracteres en el resumen corto (short_summaries) tiene en promedio 500 caracteres, el rango principal de los textos es de 400 a 700 caracteres por lo que se tiene resúmenes cortos condensados y equilibrados cumpliendo con el objetivo de tener una limitación de tokens para poder entrenar el modelo mt5-small .Por otro lado en la distribución de longitud del resumen (summary) tiene un rango más amplio de 500 a

3500 caracteres lo que indica que este texto cuenta con una estructura más detallada y extensa lo que resulta ventajoso para poder realizar un entrenamiento supervisado con el modelo mt5-small.

Figura 21

Distribución temporal por año del corpus

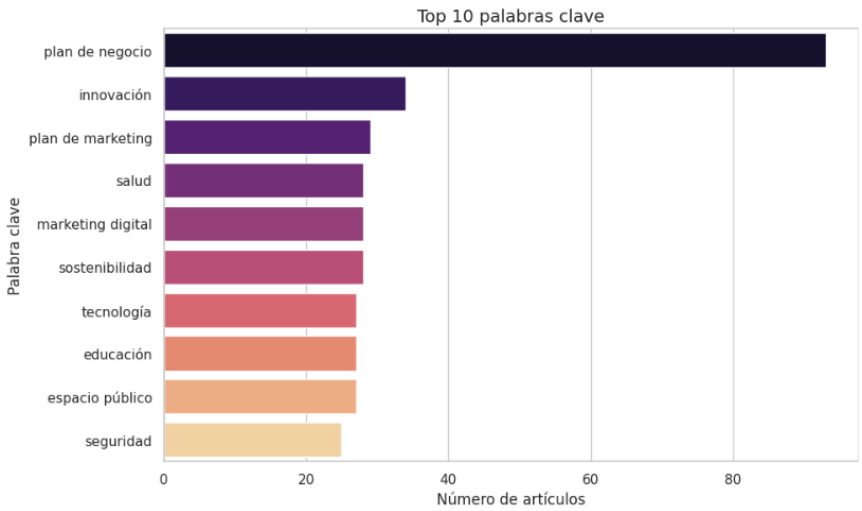


Nota: Artículos académicos por año que se usaron para el entrenamiento

En la Figura 21 se puede observar un incremento paulatino por año, que puede deberse a la mayor digitalización y crecimiento sostenido de producción académica científica en la institución. La cantidad menor de registros para 2025 se debe a que el año o ciclos académicos se encuentran en curso y no se tiene el resultado final de dicho año. Esto indica que se tiene textos actuales de los 5 años completos y uno en curso, lo que permite que el modelo se entrene con temáticas actualizadas y centradas desarrollos ecuatorianos.

Figura 22

Top 10 palabras clave



Nota. Hay una mayor concentración de textos referentes a planes de negocio

De acuerdo con el top 10 de palabras clave obtenidas en el corpus la temática de los textos extraídos es diversa, pero de manera general se tiene una mayor concentración de textos referentes a planes de negocio, innovación, marketing, salud, sostenibilidad y tecnología. Esto indica las últimas tendencias en cuanto a desarrollo de proyectos de grado o pregrado. Esto implica que el modelo entrenado pueda tener algún tipo de especialización a estos temas. Por lo que se debe validar el modelo con varios tipos de texto para ver su rendimiento en distintas temáticas.

Finalmente, se realizaron diversas visualizaciones descriptivas que evidenciaron la adecuada preparación del corpus. Los histogramas de longitud mostraron que las dos columnas principales (short_summaries y Summary) presentan distribuciones diferenciadas, permitiendo construir pares entrada-salida con densidades informativas complementarias, lo cual es esencial para el entrenamiento de modelos encoder-decoder como mT5. El análisis temporal reveló un

pico en la producción documental durante el año 2024, lo que garantiza la inclusión de textos recientes. Asimismo, se normalizó semánticamente el campo Keys, identificando las temáticas predominantes del corpus —entre ellas, planes de negocio e innovación— lo que sugiere la necesidad de validar el rendimiento del modelo con textos de diversa índole. Este conjunto de procesos permitió conformar una base lingüísticamente homogénea y estructuralmente robusta para el sistema de síntesis automática.

3.6.2. Entrenamiento (fine tuning parcial)

Empleando el corpus previamente limpiado y filtrado compuesto por pares entrada-salida estructurados como Summary (resumen original largo) y short_summaries (resumen condensado semi-automático). Se implementó un esquema de *fine-tuning parcial* sobre TPU, congelando completamente las capas del encoder y la mayoría del decoder, salvo las dos capas finales y la capa generadora (lm_head), con el fin de reducir el costo computacional y preservar el conocimiento general preentrenado del modelo. A continuación, se presenta en la Figura 23 la carga y congelamiento parcial de la arquitectura del modelo mt5-small.

Figura 23

Congelamiento parcial de capas del modelo mt5-small.

```python
# Cargar modelo con fine-tuning parcial
from transformers import MT5Tokenizer, MT5ForConditionalGeneration

tokenizer = MT5Tokenizer.from_pretrained("google/mt5-small")
model = MT5ForConditionalGeneration.from_pretrained("google/mt5-small")

# Congelar toda la arquitectura
for param in model.parameters():
    param.requires_grad = False

# Descongelar últimas 2 capas del decoder
for i, layer in enumerate(model.decoder.block):
    if i >= len(model.decoder.block) - 2:
        for param in layer.parameters():
            param.requires_grad = True

# Descongelar la capa final generadora
for param in model.lm_head.parameters():
    param.requires_grad = True

model.to(device)
```

Nota. Se cargó y congeló parcialmente la arquitectura del modelo mt5-small

A continuación, el corpus fue dividido en un conjunto de entrenamiento (70%) y otro de validación (30%) mediante train-test Split utilizando una partición estratificada controlada random_state = 99 para que pueda ser replicable y tener una base que permita evaluar el rendimiento del modelo entrenado. Para adaptar el corpus al modelo, se diseñó una clase personalizada SummarizationDataset que convierte cada par entrada-salida en tensores compatibles con la arquitectura encoder-decoder.

El campo Summary funciona como entrada principal precedida del prompt "summarize:", mientras que short_summaries representa la salida esperada. Ambos textos se tokenizan con longitudes máximas definidas (1024 para la entrada, 384 para la salida), aplicando truncamiento y padding. Se reemplazan los tokens de relleno con -100 en la salida, siguiendo las recomendaciones para el cálculo de pérdida en la librería transformers. En la Figura 24 se puede evidenciar la clase desarrollada para el preprocesamiento.

Figura 24

Código de preprocesamiento personalizado

```python
# Preprocesamiento personalizado

max_input_length = 1024
max_target_length = 384

class SummarizationDataset(Dataset):
    def __init__(self, dataframe):
        self.df = dataframe

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        source = "summarize: " + self.df.iloc[idx]["Summary"]
        target = self.df.iloc[idx]["short_summaries"]

        input_enc = tokenizer(
            source,
            max_length=max_input_length,
            padding="max_length",
            truncation=True,
            return_tensors="pt"
        )
        target_enc = tokenizer(
            target,
            max_length=max_target_length,
            padding="max_length",
            truncation=True,
            return_tensors="pt"
        )

        input_ids = input_enc["input_ids"].squeeze()
        attention_mask = input_enc["attention_mask"].squeeze()
        labels = target_enc["input_ids"].squeeze()
        labels[labels == tokenizer.pad_token_id] = -100

        return {
            "input_ids": input_ids,
            "attention_mask": attention_mask,
            "labels": labels
        }

train_df, val_df = train_test_split(df, test_size=0.3, random_state=99)
train_dataset = SummarizationDataset(train_df)
val_dataset   = SummarizationDataset(val_df)
```

Nota. Clase desarrollada para el preprocesamiento.

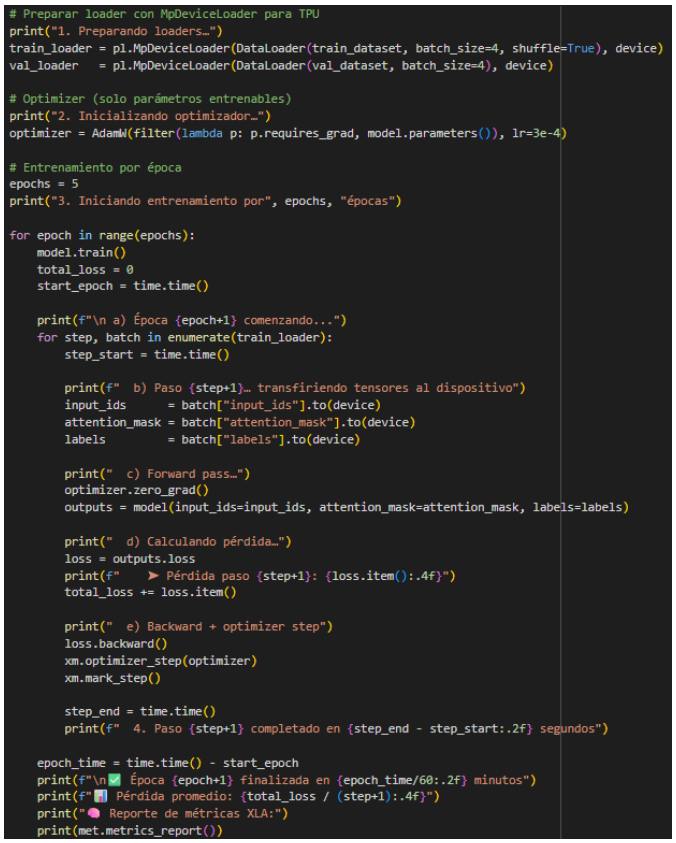
Después, el entrenamiento se realizó sobre cinco épocas utilizando el optimizador AdamW, con una tasa de aprendizaje de 3e-4.

Cada ejemplo se procesó con una longitud máxima de entrada de 1024 tokens y una salida de 384 tokens, utilizando summarize: como prompt inicial. El entrenamiento se ejecutó con torch_xla sobre TPU, aprovechando la paralelización mediante MpDeviceLoader. Durante

cada paso, se calcularon las pérdidas y se ejecutaron actualizaciones de parámetros únicamente sobre las capas descongeladas. Al finalizar el proceso, el modelo y el tokenizer ajustado se guardaron en el entorno de Google Drive para su posterior uso en tareas de resúmenes científicos automatizados. En el siguiente bloque de código (Figura 25) se puede observar el entrenamiento por época.

Figura 25

Entrenamiento por época



```
# Preparar loader con MpDeviceLoader para TPU
print("1. Preparando loaders…")
train_loader = pl.MpDeviceLoader(DataLoader(train_dataset, batch_size=4, shuffle=True), device)
val_loader   = pl.MpDeviceLoader(DataLoader(val_dataset, batch_size=4), device)

# Optimizer (solo parámetros entrenables)
print("2. Inicializando optimizador…")
optimizer = AdamW(filter(lambda p: p.requires_grad, model.parameters()), lr=3e-4)

# Entrenamiento por época
epochs = 5
print("3. Iniciando entrenamiento por", epochs, "épocas")

for epoch in range(epochs):
    model.train()
    total_loss = 0
    start_epoch = time.time()

    print(f"\n a) Época {epoch+1} comenzando...")
    for step, batch in enumerate(train_loader):
        step_start = time.time()

        print(f"  b) Paso {step+1}… transfiriendo tensores al dispositivo")
        input_ids      = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels         = batch["labels"].to(device)

        print("  c) Forward pass…")
        optimizer.zero_grad()
        outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels)

        print("  d) Calculando pérdida…")
        loss = outputs.loss
        print(f"    ➤ Pérdida paso {step+1}: {loss.item():.4f}")
        total_loss += loss.item()

        print("  e) Backward + optimizer step")
        loss.backward()
        xm.optimizer_step(optimizer)
        xm.mark_step()

        step_end = time.time()
        print(f"  4. Paso {step+1} completado en {step_end - step_start:.2f} segundos")

    epoch_time = time.time() - start_epoch
    print(f"\n✅ Época {epoch+1} finalizada en {epoch_time/60:.2f} minutos")
    print(f"📊 Pérdida promedio: {total_loss / (step+1):.4f}")
    print("🧠 Reporte de métricas XLA:")
    print(met.metrics_report())
```

Nota. Entrenamiento del Modelo.

3.6.3. Evaluación del Modelo

Finalmente, con el modelo entrenado y para evaluar su rendimiento, se calcularon métricas de tipo ROUGE sobre una muestra representativa del conjunto de validación, es decir alrededor del 10 % de la base de validación, 50 registros aleatorios.

Las métricas de ROUGE permiten comparar de forma automática el resumen generado por el modelo con el texto completo para validar coincidencia léxica y estructural. Se consideró la métrica de ROUGE-1, ROUGE-2, ROUGE-L y ROUGE-Lsum para tener una visión integral de la cobertura, calidad estructural, coherencia y calidad sintáctica del resumen generado por el modelo entrenado con fine tuning parcial.

A continuación, se detallan los valores obtenidos y sus implicaciones dentro del marco de generación automática de resúmenes académicos.

- ROUGE-1 (0.1492): El 14.92% del resumen generado coincide con el texto de referencia considerando palabras. Esto implica una cobertura básica o superficial del texto completo.
- ROUGE-2 (0.0000): No existen coincidencias de pares de palabras entre el texto y el resumen generado. Esto implica problemas estructurales y poca fluidez del resumen generado.
- ROUGE-L (0.1361): El 13.61% del resumen generado tiene una similitud estructural con el texto completo. Esto implica una evaluación de coherencia básica en el resumen generado.
- ROUGE-Lsum (0.1369): El 13.69% mide la coherencia entre frases completas, lo cual es útil para textos largos.

Considerando los resultados obtenidos en las cuatro métricas, se puede decir que el modelo entrenado con fine tuning parcial tiene un nivel básico de cobertura léxica porque el 15% aproximadamente de las palabras coincide con el texto original, hay margen de mejora en términos de fluidez semántica y estructura general de las oraciones.

3.7. Infraestructura en la Nube

Todo el sistema fue montado en los servicios de Google Cloud Platform (GCP), lo que permitió que funcionara de forma estable, rápida y segura. Usar la nube hizo posible procesar varios documentos al mismo tiempo, guardar la información sin riesgo de pérdida, y además tener la flexibilidad para crecer si el proyecto necesita manejar más carga o usuarios en el futuro.

3.7.1. Servicios utilizados

Los principales servicios de GCP empleados fueron:

- Google Cloud Vision API: que se encargó de convertir imágenes de texto (como las que vienen en los PDFs escaneados) en texto editable. Esta herramienta es muy precisa, funciona en varios idiomas y ofrece resultados rápidos.
- Google Cloud Storage (GCS): donde se guardaron todos los archivos del sistema: desde los documentos originales hasta los textos procesados y los resúmenes generados. Este servicio permite controlar quién puede acceder a qué archivos, y también llevar un control de versiones si se necesita.

3.7.2. Ventajas de la infraestructura en la nube

El uso de GCP trajo múltiples beneficios:

- Escalabilidad: se puede procesar más de un documento en paralelo si se incrementa la capacidad de los servicios.
- Disponibilidad: los servicios de Google garantizan disponibilidad del 99.9%, lo cual asegura continuidad operativa.
- Seguridad: mediante el uso de autenticación de claves, IAM y registros de acceso, se protegen los datos y modelos involucrados.
- Facilidad de mantenimiento: la centralización en la nube evita depender de servidores locales, simplificando el monitoreo y las actualizaciones del sistema.

CAPITULO 4

4. ANÁLISIS DE RESULTADOS

4.1. Pruebas de Concepto

Como parte del proceso de validación, se llevaron a cabo pruebas de concepto con el objetivo de observar el comportamiento y desempeño de distintos modelos de lenguaje ante tareas específicas de comprensión y resumen de textos científicos. Estas pruebas permitieron evaluar si la herramienta desarrollada cumplía con los criterios previamente definidos y detectar posibles áreas de mejora.

Se utilizaron varios modelos disponibles los cuales fueron seleccionados por su disponibilidad y relevancia actual en el campo del procesamiento de lenguaje natural, entre ellos:

- Mistral NEMU
- LLaMA 3.3
- Kimi VL
- MT5 – Small (fine tuning parcial)

Los usuarios interactuaron con la herramienta mediante la carga de archivos PDF, sobre los cuales podían generar resúmenes automáticos o formular preguntas específicas relacionadas con el contenido del documento.

A continuación, se indica la vista general del sistema de generación automática de resúmenes científicos con modelos NLP desarrollado:

El aplicativo desarrollado presenta una interfaz completamente intuitiva y amigable para el usuario. Al iniciar, se muestra una pantalla principal con una breve explicación sobre las funciones de la herramienta, permitiendo al usuario comprender desde el principio cuál es su propósito y cómo utilizarla tal como lo refleja la Figura 26.

Figura 26

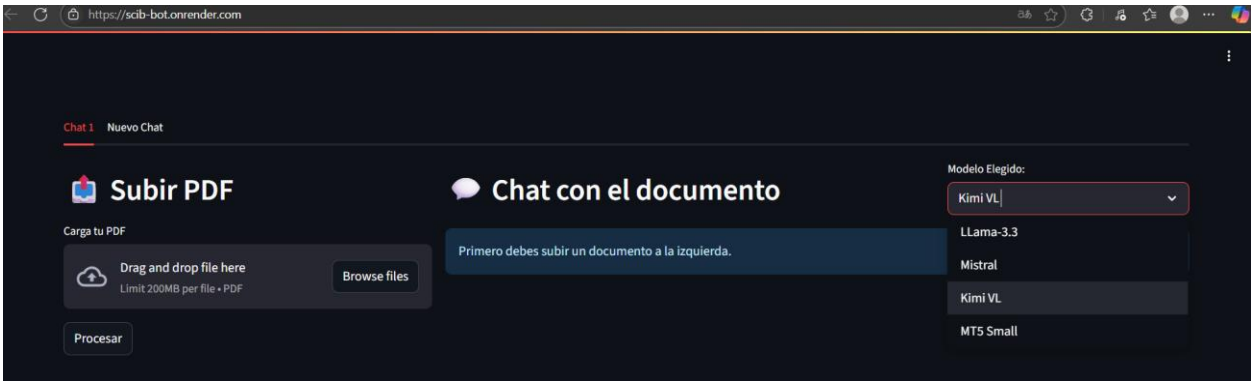
Pantalla General

Nota. Corresponde a la pantalla que verá el usuario.

Una vez que se hace clic en la opción "Iniciar programa", se despliega una nueva ventana en la que es posible seleccionar el modelo de lenguaje con el que se desea trabajar. Esta funcionalidad permite comparar el comportamiento de diferentes modelos y analizar su desempeño bajo las mismas condiciones como se lo aprecia en la Figura 27.

Figura 27

Tipos de Modelos

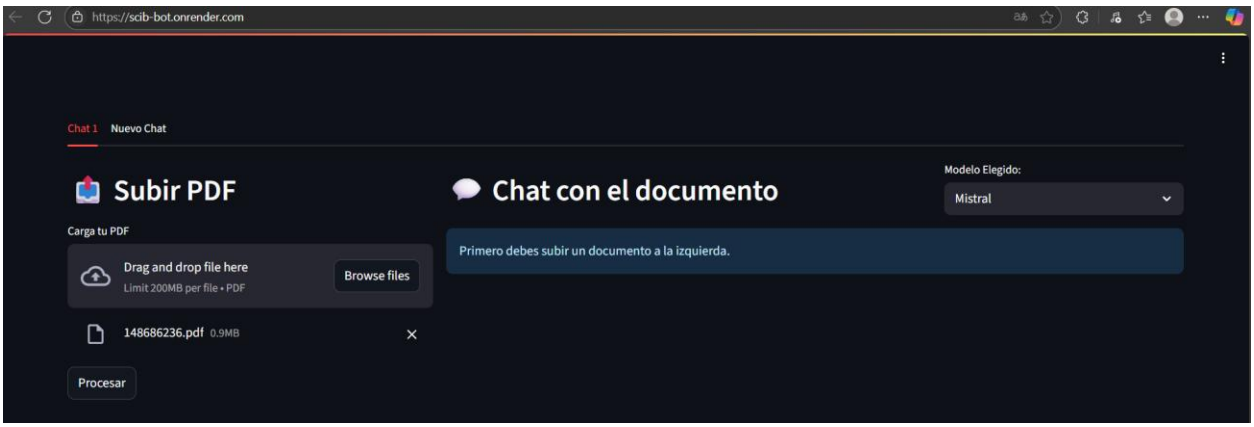


Nota. El usuario puede seleccionar el modelo de su preferencia

A continuación, el usuario puede cargar un archivo en formato PDF mediante el botón "Browse File". Una vez seleccionado el documento, se debe hacer clic en "Procesar" para que la herramienta inicie el análisis del contenido. El sistema procesa automáticamente el texto y genera un resumen basado en la información contenida en el documento.

Figura 28

Procesamiento de la Información

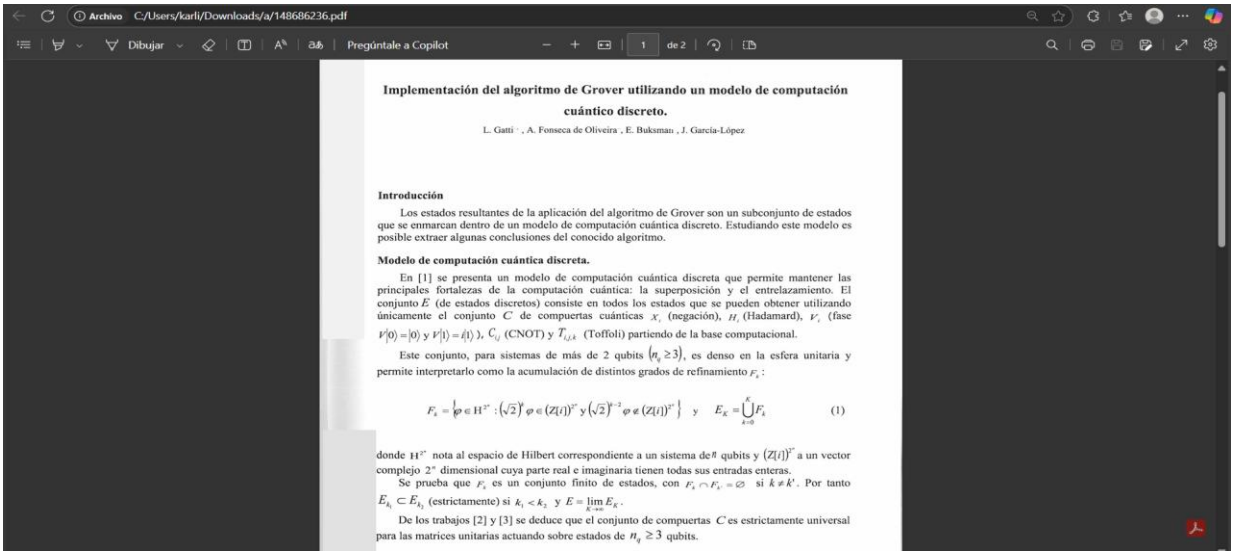


Nota. Una vez seleccionado y enviado el archivo PDF empieza el procesamiento de la información

Además de la función de resumen, la herramienta permite realizar preguntas de manera libre, simulando una conversación tipo ChatBot. Esta característica permite una interacción dinámica con el contenido del PDF, facilitando la comprensión de textos complejos o extensos sin necesidad de leerlos por completo.

Figura 29

Texto para probar



Nota. Corresponde al texto que va a ser procesado y del cual se realizarán las preguntas

Desde el diseño, se tomaron medidas para mejorar la precisión de las respuestas. En particular, se configuró una temperatura baja en los modelos generativos, lo que reduce la probabilidad de que el sistema proporcione respuestas inventadas o irrelevantes (alucinaciones)

Gracias a esta configuración, el modelo se mantiene estrictamente limitado al contenido del documento cargado, sin ofrecer información que no esté respaldada por el texto original.

En el Anexo I se presenta de forma más exhaustiva y detallada el proceso de prueba realizado con los cuatro modelos evaluados, junto con las respuestas generadas por cada uno.

Las pruebas incluyeron tanto preguntas abiertas como instrucciones directas, con el objetivo de valorar la capacidad de los modelos para interpretar el contenido del documento PDF cargado, extraer información precisa y responder con coherencia.

Las entradas utilizadas en esta fase fueron las siguientes:

- Hola
- ¿Quiénes son los autores del documento?
- ¿Quién es Gustavo Adolfo Becker?
- ¿Cuál es la idea principal del documento?
- ¿Cómo citar en normas APA el documento?

4.2. Análisis de Resultados

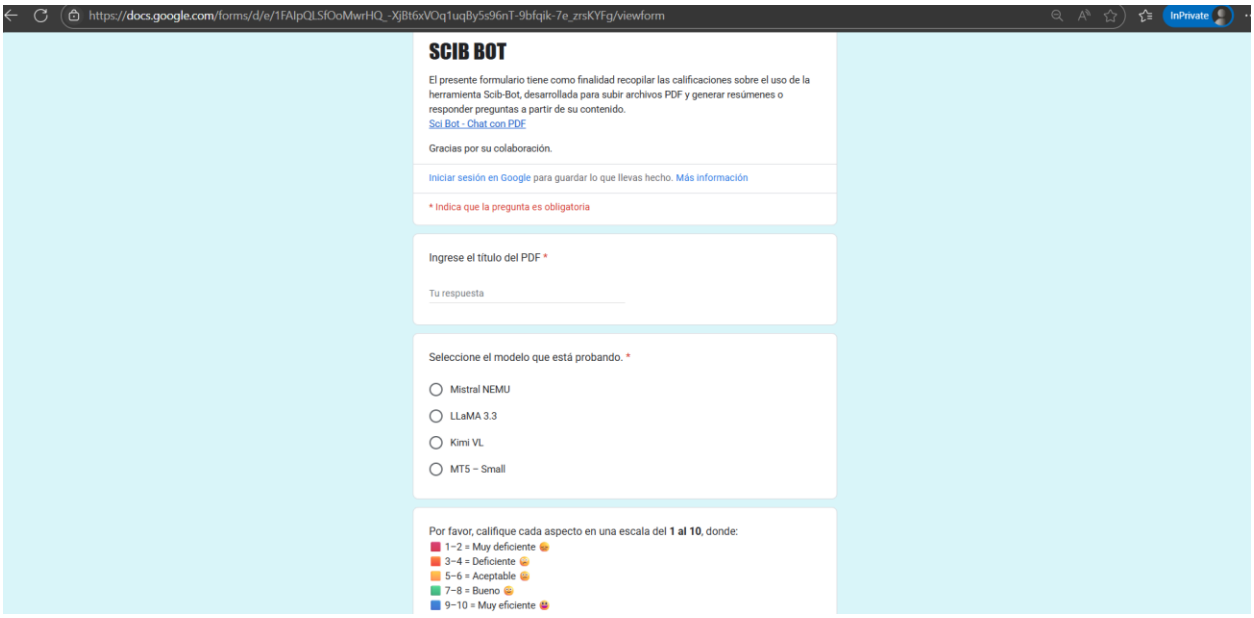
Para evaluar el rendimiento del modelo de resumen, se diseñó y aplicó un formulario (Google Forms) basado en cuatro criterios principales (Ver figura 30):

- Calidad de la respuesta
- Adaptación al tipo de contenido

- Funcionamiento técnico
- Aspectos éticos y sociales.

Figura 30

Formulario de Evaluación de Modelos



Nota. Formulario enviado: SCIB BOT Aplicación a probar por los usuarios: Streamlit

El formulario fue enviado a un grupo de evaluadores que analizaron el desempeño del modelo respondiendo a las preguntas establecidas. Las preguntas fueron:

1. Calidad de la respuesta

Relevancia: Los evaluadores consideraron si el resumen destacaba adecuadamente las ideas clave del texto original.

Claridad: Se valoró que la respuesta fuera coherente con el contenido, evitando ambigüedades y manteniendo el contexto.

Coherencia: Se analizó la capacidad del modelo para reconocer preguntas fuera de contexto y alertar al usuario.

Precisión / Exactitud: Se verificó si la información proporcionada era correcta, precisa y verificable.

2. Adaptación al tipo de contenido

Adaptación científica: Se evaluó si el modelo respetaba el tono y lenguaje propios de un artículo científico.

Multilenguaje: Se revisó la calidad de las respuestas en diferentes idiomas, garantizando consistencia.

3. Funcionamiento técnico

Eficiencia computacional: Se analizó el tiempo de respuesta y si este cumplía con estándares razonables sin afectar la calidad.

4. Aspectos éticos y sociales

Inclusión: Se valoró la ausencia de sesgos relacionados con género, raza u otros aspectos culturales o sociales.

Privacidad: Se confirmó que el modelo respetara la privacidad, evitando revelar información sensible.

Ética general: Se comprobó que las respuestas respetaran principios éticos, evitando contenido ofensivo o dañino.

4.3.Análisis de Resultados

La tabla 6 indica las 112 respuestas del formulario de Google de la evaluación de la aplicación. La cantidad promedio de respuestas generadas por cada modelo fue de 28 archivos PDF. El orden de puntuación de los modelos fue:

1. LLaMA 3.3
2. Kimi VL
3. Mistral NEMU
4. MT5 – Small

Tabla 6

Resultados de Evaluación

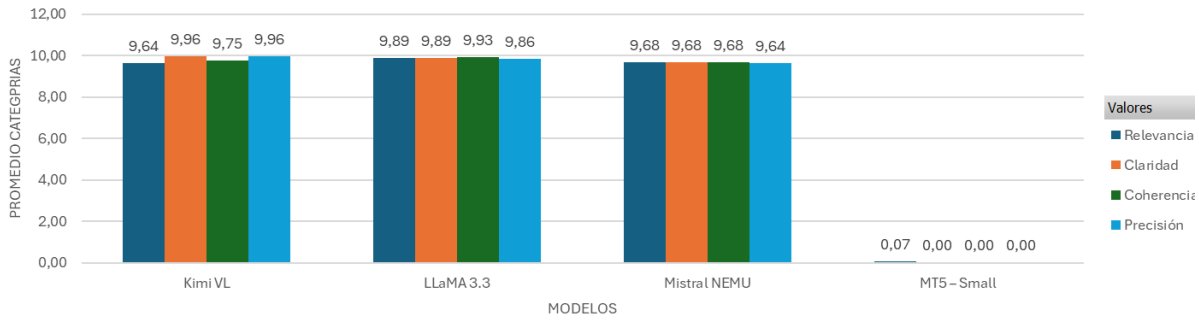
Modelos	**Cantidad PDF**	**Promedio de Categorías**
Kimi VL	28	9,90
LLaMA 3.3	28	9,91
Mistral NEMU	28	9,65
MT5 – Small	28	0,01
Total General	**112**	

Nota. La calidad de la respuesta también depende de la calidad de la pregunta.

En la categoría calidad de la respuesta el Modelo mejor puntuado fue LLaMA 3.3 como se lo aprecia en la Figura 31 con una puntuación casi perfecta.

Figura 31

Calidad de la respuesta

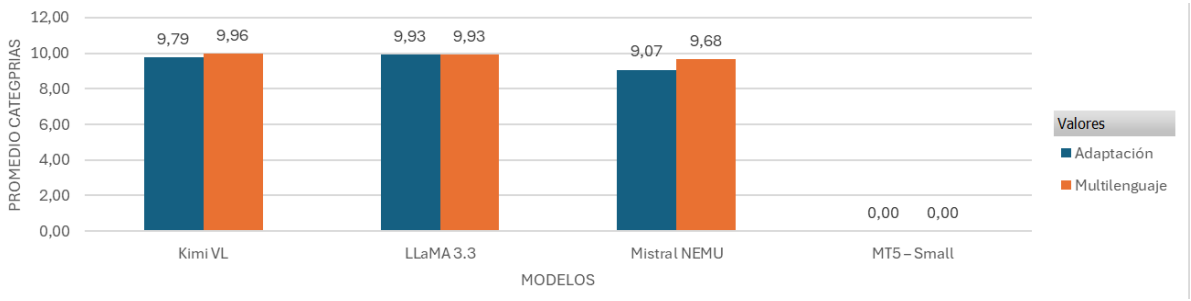


Nota: Resultado a partir de las 112 respuestas de la encuesta.

En la categoría Adaptación al tipo de contenido el Modelo mejor puntuado fue nuevamente LLaMA 3.3 como lo indica la Figura 32.

Figura 32

Adaptación al tipo de contenido

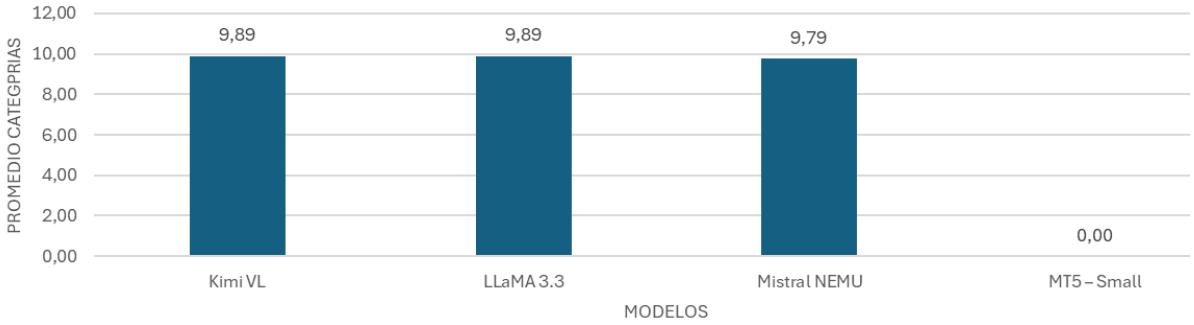


Nota: Resultado a partir de las 112 respuestas de la encuesta.

En la categoría Funcionamiento técnico el Modelo mejor puntuado fue nuevamente LLaMA 3.3 como se refleja en la Figura 33 con un promedio de 9.89

Figura 33

Funcionamiento técnico

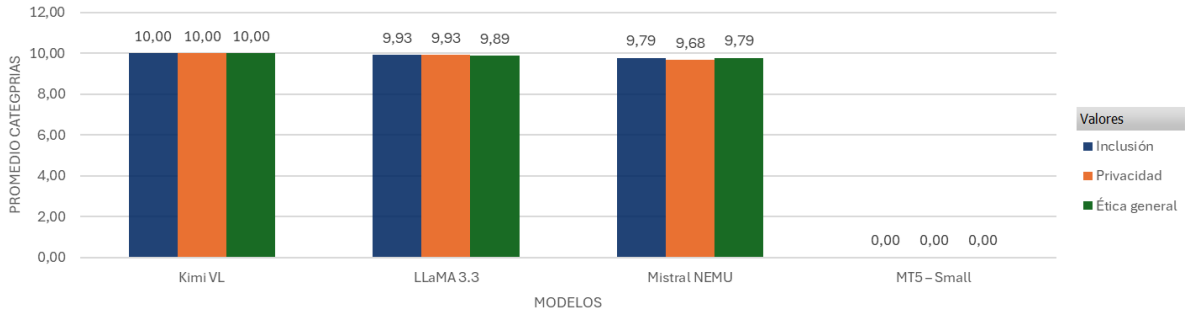


Nota: Resultado a partir de las 112 respuestas de la encuesta.

La Figura 34 indica la categoría Aspectos éticos y sociales, donde el Modelo mejor puntuado fue LLaMA 3.3

Figura 34

Aspectos éticos y sociales



Nota: Resultado a partir de las 112 respuestas de la encuesta.

CAPITULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

Se desarrolló una herramienta multifuncional basada en modelos OCR, procesamiento de lenguaje natural y un chatbot resulta muy útil para quienes inician un proceso de investigación. Al agrupar en una sola aplicación funciones como la extracción de texto desde distintos tipos de documentos, la generación de resúmenes claros y la posibilidad de explorar temas específicos según la necesidad del investigador, misma que logra ahorrar tiempo y mejorar la organización de la información.

Adicionalmente, tras las pruebas realizadas, se comprobó que el modelo LLaMA 3.3 integrado en el sistema fue el modelo que más cumple con el objetivo principal del proyecto, ya que entrega resúmenes bien estructurados con lenguaje técnico, ideal para trabajos científicos. Su versión gratuita también permite hacer más consultas, lo que ayuda al investigador a decidir con rapidez qué documentos puede profundizar y cuáles pueden descartarse. En conjunto, todo esto hace que el proceso de investigación sea más ágil, efectivo y enfocado.

5.1.1. Conclusiones Backend

El diseño del back-end y la estructura de modelos resultaron adecuados para soportar la funcionalidad principal del sistema; sin embargo, la dependencia de servicios de terceros bajo planes gratuitos limita no solo el rendimiento, sino también la continuidad del desarrollo.

La integración de modelos de lenguaje en el back-end fue funcional y efectiva para generar resúmenes científicos. Sin embargo, surgen efectos distantes cuando se intenta trabajar

con modelos open source high-profile: las demandas computacionales superan las capacidades ofrecidas por planes gratuitos, afectando la viabilidad del sistema.

Se utilizó el modelo MT5 Small con un fine-tuning parcial para la tarea de resumen, pero los resultados no fueron del todo adecuados. Esto puede atribuirse tanto al entrenamiento limitado del modelo, posiblemente debido a la escasez de datos disponibles como a la insuficiencia de recursos computacionales para llevar a cabo un entrenamiento más completo. Estas limitaciones condicionan la calidad del modelo generado y, en consecuencia, comprometen la efectividad general del sistema.

5.1.2. Conclusiones Frontend

Streamlit permite avanzar rápido sin complicarse, ya que con solo Python se puede crear una interfaz funcional sin necesidad de saber desarrollo web.

Streamlit se integra fácilmente con modelos de inteligencia artificial, lo que facilita mostrar resultados en tiempo real sin procesos complicados.

Streamlit ofrece una interfaz clara y sencilla para usuarios no técnicos, lo que hace más accesible el sistema para cualquier persona.

La selección de modelos facilita probar y comparar diferentes modelos de IA en tiempo real, lo cual es útil para evaluar su rendimiento en un mismo documento.

5.2. Recomendaciones

Considerando los resultados obtenidos, se recomienda para futuras investigaciones se continue con el uso de modelos de lenguaje ya pre-entrenados, dado que esto han demostrado ser más eficientes y confiables para generar resúmenes y realizar tareas de comprensión de textos científicos. Estos modelos, al estar entrenados con grandes volúmenes de datos y respaldados por una infraestructura robusta, ofrecen resultados de mejor calidad sin necesidad de un proceso de entrenamiento adicional.

Antes de implementar técnicas de entrenamiento profundo, se recomienda evaluar cuidadosamente la disponibilidad de infraestructura computacional y la cantidad y calidad de los datos de entrenamiento. Dadas las limitaciones técnicas encontradas en este estudio, no fue posible aplicar un fine-tuning completo a los modelos seleccionados, por lo que se optó por realizar comparaciones utilizando prompts específicos. No obstante, si en futuras etapas del proyecto se cuenta con mejores recursos y una mayor disponibilidad de datos, retomar el fine-tuning podría representar una alternativa válida para personalizar y mejorar el rendimiento del modelo en tareas específicas.

5.2.1. Recomendaciones Backend

Se recomienda reducir la dependencia de servicios gratuitos integrando modelos open source autoalojados, lo que brinda mayor control y adaptabilidad. En caso de contar con presupuesto, también es viable optar por planes pagos de servicios externos que ofrezcan mejor rendimiento, estabilidad y soporte, garantizando la continuidad y escalabilidad del sistema.

Se recomienda migrar hacia infraestructura más potente, como servidores propios, procesamiento en la nube o modelos optimizados que permitan sostener el rendimiento sin depender de recursos limitados.

Se recomienda realizar un fine-tuning más profundo del modelo MT5 Small u optar por un modelo más robusto, siempre que los recursos lo permitan. Es fundamental asegurar una mayor disponibilidad de datos representativos y de calidad para el entrenamiento, así como contar con infraestructura computacional adecuada.

5.2.2. Recomendaciones Frontend

Se recomienda separar la interfaz del resto de la lógica en el código, para mantener todo más ordenado, fácil de mantener y con una mejor escalabilidad.

Se recomienda que el flujo de uso de la aplicación sea lo más simple y guiado posible, para facilitar su uso por parte de usuarios sin conocimientos técnicos.

6. REFERENCIAS BIBLIOGRÁFICAS

Bellman, R. (1978). *An Introduction to Artificial Intelligence: Can Computers Think?* San Francisco: Boyd & Frase Publishing Company.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., & Amodei, D. (2020). *Language models are few-shot learners. Advances in Neural Information Processing Systems*. Obtenido de https://arxiv.org/abs/2005.14165

Charniak, E., & Drew. (1985). *Introduction to Artifical Intellgence* . Massachusetts: Addison-Wesley.

Diab, M., Herrera, J., Sleep, M., Chernow, B., & & Mao, C. (21 de Abril de 2024). *Stable Diffusion Prompt Book.* Obtenido de OpenArt.: https://cdn.openart.ai/assets/Stable%20Diffusion%20Prompt%20Book%20From%20OpenArt%2011-13.pdf

Diao, S., Wang, P., Lin, Y., & Zhang, T. (2023). *Active Prompting with Chain-of-Thought for Large Language Models* . Obtenido de Arxiv: https://arxiv.org/abs/2302.12246

Díaz Subieta, L. B. (2024). El uso de la inteligencia artificial en la investigación científica. *Revista Historia de la Educación Latinoamericana, 26*(43). doi:https://doi.org/10.19053/uptc.01227238.18014

Fu, Y., Peng, H., Sabharwal, A., Clark, P., & y Khot, T. (2022). *Complexity-Based Prompting for Multi-Step Reasoning*. Obtenido de https://arxiv.org/abs/2210.00720

Ganesan, M., & Devi, S. (2025). *Automated PDF Data Extraction and Retrieval Using NLP and OCR. International Journal of Innovative Research in Technology*. Obtenido de Ijirt: https://ijirt.org/publishedpaper/IJIRT175285_PAPER.pdf

God of Prompts. (2024). *Understanding the Basics of Prompt Design*. Obtenido de https://godmodechatgpt.notion.site/Understanding-the-Basics-of-Prompt-Design-4279de4f02984c41817b556443605b64

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* Cambridge, Massachusetts: MIT Press.

Gordon, P. (2019). Information overload in the information age: a review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development. *Business Research*, 479-522. doi: https://doi.org/10.1007/s40685-018-0069-z

Haugeland, J. (1985). *Artificial Intelligence: The Very Idea.* Massachusetts: MIT Press.

Hong, Z., Ward, L., Chard, K., Blaiszik, B., & Foster, I. (2021). *Challenges and Advances in Information Extraction From Scientific Literature: A Review*. Obtenido de https://doi.org/10.1007/s11837-021-04902-9

Howard, J., & Ruder, S. (2018). *Universal Language Model Fine-tuning for Text Classification*. Obtenido de arXiv: https://arxiv.org/abs/1801.06146

huggingface. (2025). *huggingface.co*. Obtenido de huggingface: https://huggingface.co/google/mt5-small#abstract

Jung, J., Qin, L., Welleck, S., Brahman, F., Bhagavatula, C., Le Bras, R., & Choi, Y. (2022). *Maieutic Prompting: Logically Consistent Reasoning with Recursive Explanations*. Obtenido de https://arxiv.org/abs/2205.11822

Kumar, S., & Solanki, A. (2023). *An abstractive text summarization technique using transformer model with self-attention mechanism*. Obtenido de Springer Nature: https://doi.org/10.1007/s00521-023-08687-7

Kurzweil, R. (1990). *The Age of Intelligent Machine.* Massachusetts: Cambridge.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, págs. (436–444).

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . D., K. (2020). *Retocerieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In Proceedings Advances in Neural Information Prssing Systems 33 (NeurIPS 2020).* Obtenido de https://dl.acm.org/doi/10.5555/3495724.3496517

Li, Z., Peng, B., He, P., Galley, M., Gao, J., & y Yan, X. (2023). *Guiding Large Language Models via Directional Stimulus Prompting*. Obtenido de https://arxiv.org/abs/2302.11520

Liu, J., Liu, A., Lu, X., Welleck, S., West, P., Le Bras, R., . . . Hajishirzi, H. (2022). *Generated Knowledge Prompting for Commonsense Reasoning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Dublin, Ireland: Association for Computational Linguistics.

Llamaindex. (2024). *How each index works*. Obtenido de https://docs.llamaindex.ai/en/v0.10.17/module_guides/indexing/index_guide.html#h ow-each-index-works

López Takeyas, B. (2007). *Introducción a la Inteligencia Artificial*. Obtenido de Instituto Tecnológico de Nuevo Laredo: https://www.utm.mx/~jjf/ia/A1.pdf

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *ICLR*, 1-11. Obtenido de https://arxiv.org/abs/1711.05101

McAuliffe, Z. (2022). *Google's Latest AI Model Can Be Taught How to Solve Problems.CNET*. Obtenido de https://www.technologyreview.com/2025/05/14/1116438/google-deepminds-new-ai-uses-large-language-models-to-crack-real-world-problems/

Mejía Trejo, J. (2024). *Inteligencia Artificial. Fundamentos de Ingeniería de Prompts con ChatGPT como Innovación impulsora de la Creatividad: (1 ed).* Jalisco: Academia Mexicana de Investigación y Docencia en Innovación (AMIDI).

Mitchell, R. (2015). *Web scraping with Python: Collecting more data from the modern web.* O'Reilly Media.

Nillson, N. (1998). *Artificial Intelligence: A new Synthesis.* San Mateo, California: Morgan Kaufman.

OpenAI. (2022). *Introducing ChatGPT*. Obtenido de https://openai.com/es-ES/index/chatgpt/

OpenAI. (2023). *GPT-4 Technical Report*. Obtenido de https://openai.com/es-ES/index/gpt-4-research/

OpenAI. (2024). *GPT-4o and more tools for ChatGPT Free*. Obtenido de https://openai.com/es-ES/index/gpt-4o-and-more-tools-to-chatgpt-free/

OpenAI. (2025). *GPT-4.1 Release Notes*. Obtenido de https://openai.com/index/gpt-4-1/

Perez, F., & Ribeiro, I. (2022). *Ignore Previous Prompt: Attack Techniques For Language Models*. Obtenido de https://arxiv.org/abs/2211.09527

Pisano, P. (29 de Abril de 2022). *Crítica y oportunidad de la inteligencia artificial*. Obtenido de MIT Technology Review en español: https://www.technologyreview.es/s/14195/critica-y-oportunidad-de-la-inteligencia-artificial

Poole, D., Mackworth, A., & Goebel, R. (1998). *Computational Intelligence: A logical approach.* Oxford, Reino Unido: Oxford University Press.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, H. (2023). *Language Models are Unsupervised Multitask Learners*. Obtenido de https://archive.org/details/language-models-are-unsupervised-multitask-learners

Raschka, S., Mirjalili, V., & Liu, Y. (2023). *Machine Learning con PyTorch y Scikit-Learn: (1 ed.).* Barcelona (España): Marcombo.

Rouhiainen, L. (2018). *Inteligencia Artificial. 101 Cosas que debes saber hoy y sobre nuestro Futuro.* Barcelona: Alienta.

Rubio, C. (14 de Noviembre de 2023). *Los expertos analizan el lado oscuro de la IA: «La única escapatoria del ser humano es la crítica y la duda»*. Obtenido de El Debate: https://www.eldebate.com/tecnologia/20231114/expertos-analizan-lado-oscuro-ia-unica-escapatoria-humano-critica-duda_151400.html

Rush, A. M., Chopra, S., & Weston, J. (2015). *A neural attention model for abstractive sentence summarization*. Obtenido de Arxiv: https://arxiv.org/abs/1509.00685

Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach.3rd.ed. Prentice Hall Series in Artificial intelligence.* New Jersey: Prentice Hall.

Sahoo, P., Singh, A., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2024). *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. Obtenido de https://arxiv.org/abs/2402.07927

Selvi, J. (2022). *Exploring Prompt Injection Attacks.* Obtenido de https://www.nccgroup.com/us/research-blog/exploring-prompt-injection-attacks/

Sinha, R., & B., R. (2024). Digitization of document and information extraction using OCR. (arXiv, Ed.) *RV College of Engineering*. Obtenido de https://arxiv.org/pdf/2506.11156

U.S. Government Accountability Office, G. (2022). *Consumer data: Increasing use poses risks to privacy (GAO-22-106096).* Obtenido de https://www.gao.gov/products/gao-22-106096

UNESCO. (2024). *Aplicaciones de la IA para la gestión del agua.* Obtenido de https://unesdoc.unesco.org/ark:/48223/pf0000393243

Valdivia, A., & Tazzioli, M. (27 de Noviembre de 2021). *Una crítica a la inteligencia artificial más allá de los sesgos*. Obtenido de El Salto Diario: https://www.elsaltodiario.com/paradoja-jevons-ciencia-poder/critica-inteligencia-artificial-sesgos

Valinsky, J. (11 de Abril de 2019). *Amazon reportedly employs thousands of people to listen to your Alexa conversations*. Obtenido de CNN Business: https://edition.cnn.com/2019/04/11/tech/amazon-alexa-listening/index.html

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., . . . D., Z. (2022). *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. Obtenido de https://arxiv.org/abs/2203.11171

Winston, P. (1992). *Artificial Intelligence (Third Edition).* Massachusetts: Addison-Wesley.

Xiang, C. (2023). *The Amateurs Jailbreaking GPT Say They're Preventing a Closed-Source AI Dystopia*. Obtenido de https://www.vice.com/en/article/5d9z55/jailbreak-gpt-openai-closed-source

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., . . . Raffel, C. (2020). mT5: AMassively Multilingual Pre-trained Text-to-Text Transformer. *arXiv preprint*. Obtenido de https://arxiv.org/abs/2010.11934}

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., & Narasimhan, K. (2023). *Tree of Thoughts: Deliberate Problem Solving with Large Language Models.* Obtenido de https://arxiv.org/abs/2305.10601

Zhang, D., Yin, J., & & Zhang, X. (2020). *Representation Learning for Natural Language Processing*. Obtenido de Daobooks: https://directory.doabooks.org/handle/20.500.12854/35038

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., . . . Cui, C. (2022). *Least-to-Most Prompting Enables Complex Reasoning in Large Language Models*. Obtenido de https://arxiv.org/abs/2205.10625

Ziegler, A., & Berryman, J. (17 de Julio de 2023). *A developer's guide to prompt engineering and LLMs*. Obtenido de GitHub: https://github.blog/2023-07-17-prompt-engineering-guide-generative-ai-llms/

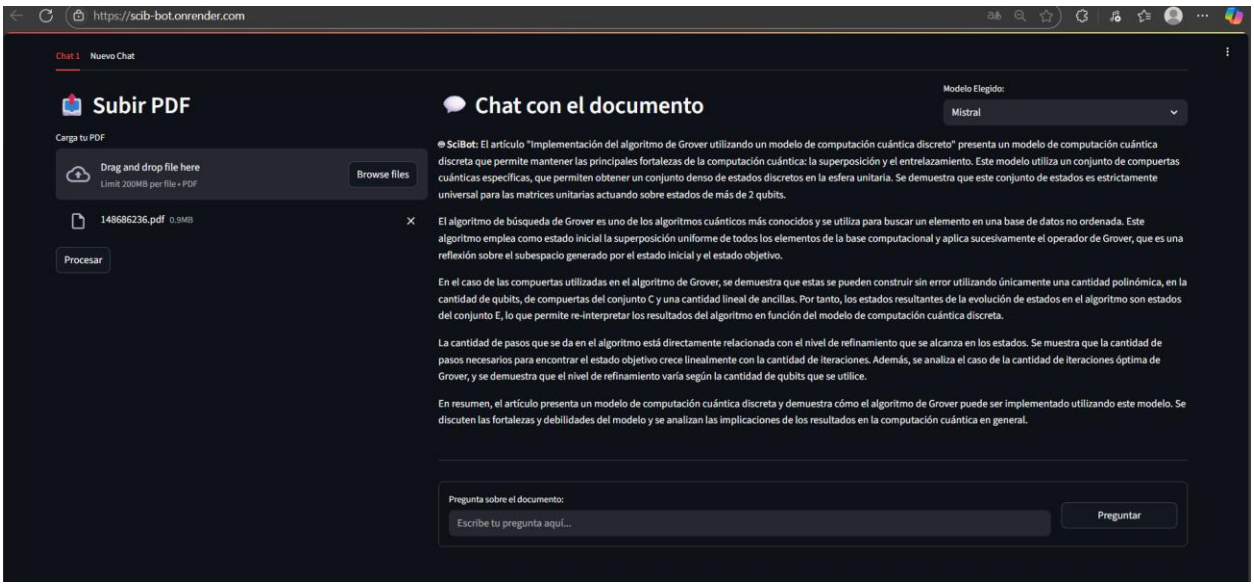
7. ANEXOS

7.1. ANEXO I

7.1.1. MODELO PROBADO: MISTRAL NEMU

Figura 35

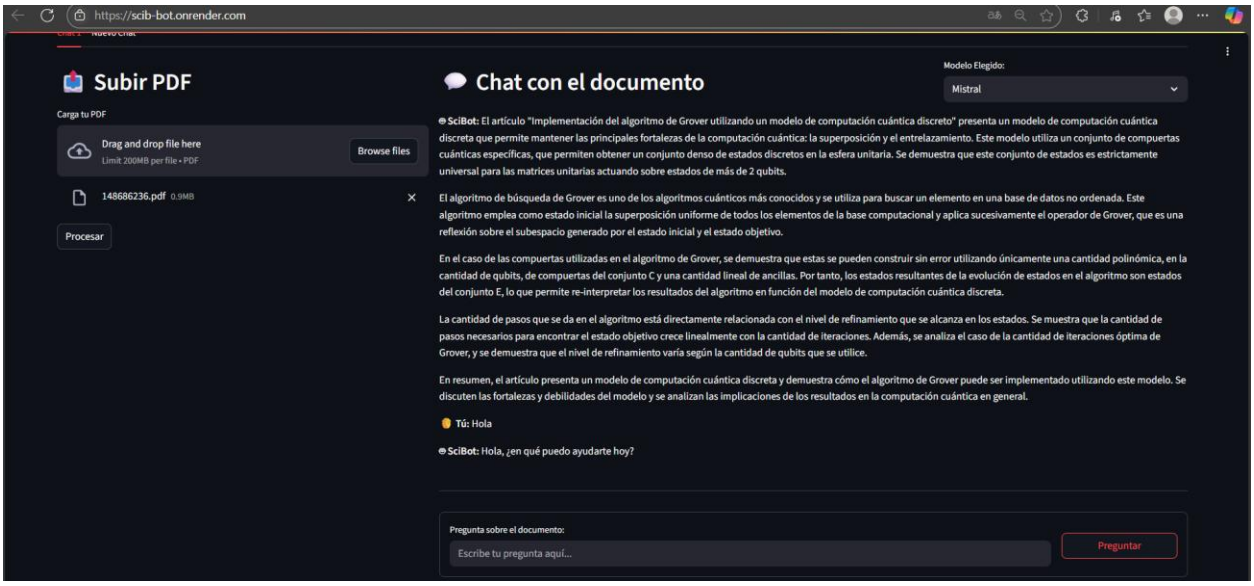
Obtención de Texto



Nota. Se carga el PDF y se lo prueba con el modelo Mistral NEMU

Figura 36

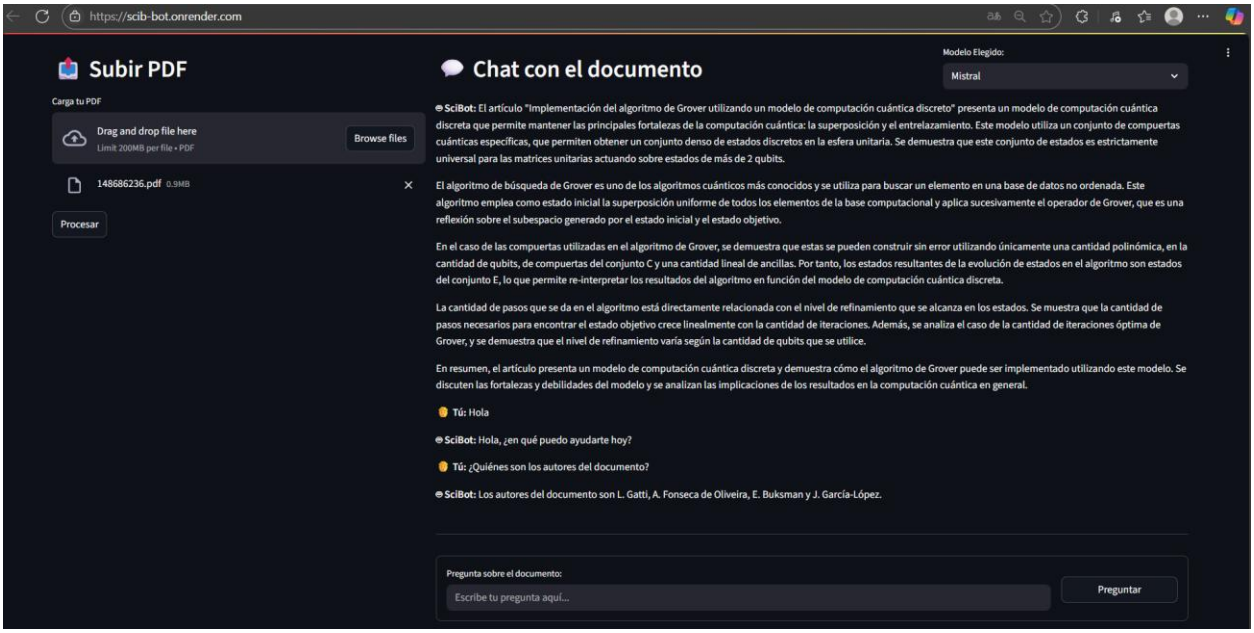
Pregunta 1 del Modelo Mistral NEMU



Nota. Se prueba la pregunta 1 con el modelo Mistral NEMU

Figura 37

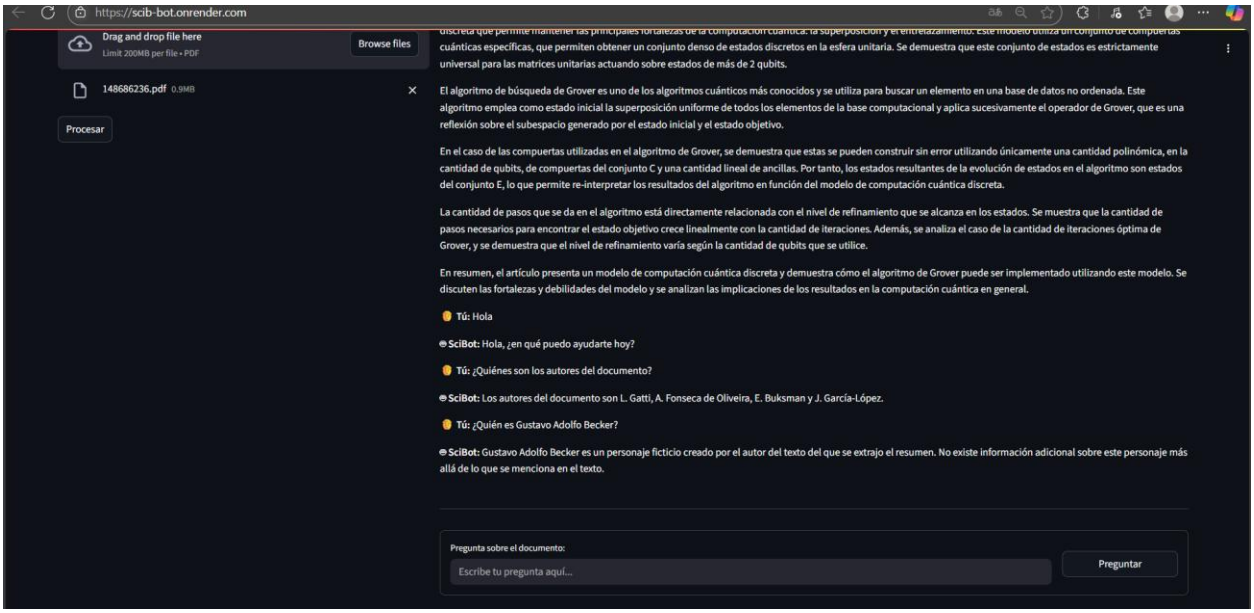
Pregunta 2 del Modelo Mistral NEMU



Nota. Se prueba la pregunta 2 con el modelo Mistral NEMU

Figura 38

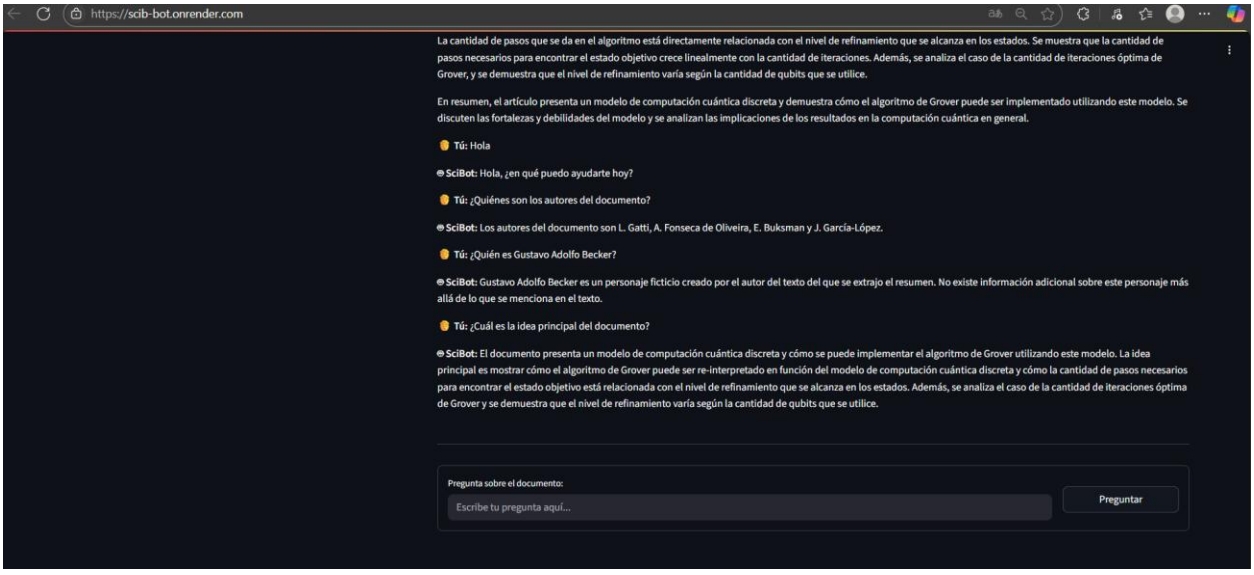
Pregunta 3 del Modelo Mistral NEMU



Nota. Se prueba la pregunta 3 con el modelo Mistral NEMU

Figura 39

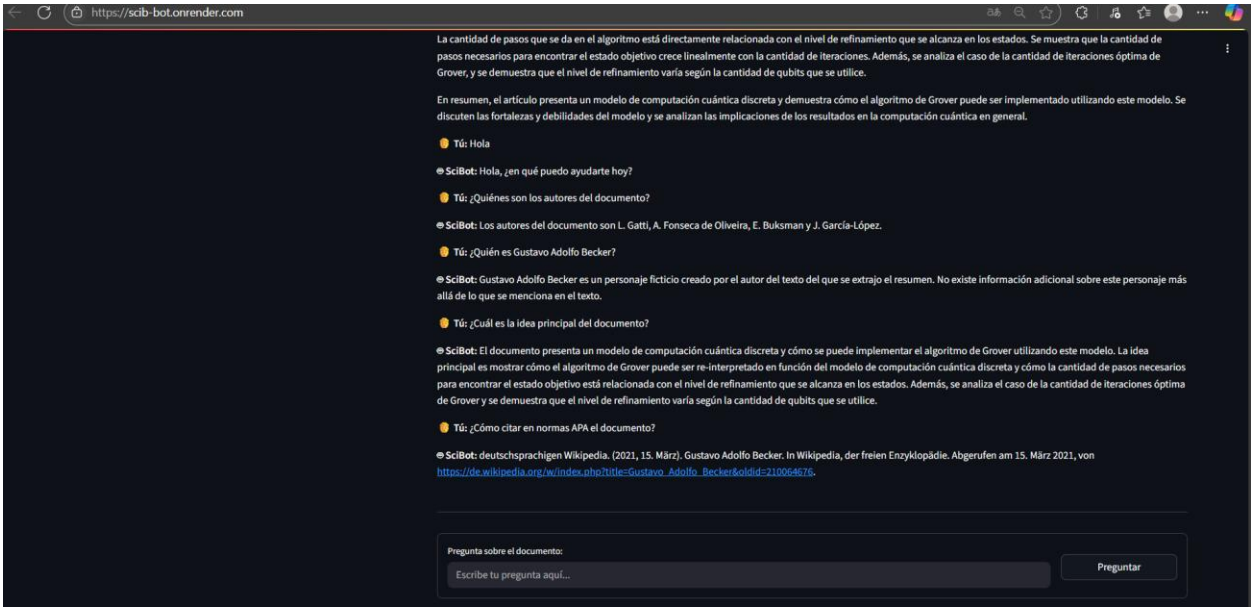
Pregunta 4 del Modelo Mistral NEMU



Nota. Se prueba la pregunta 4 con el modelo Mistral NEMU

Figura 40

Pregunta 5 del Modelo Mistral NEMU

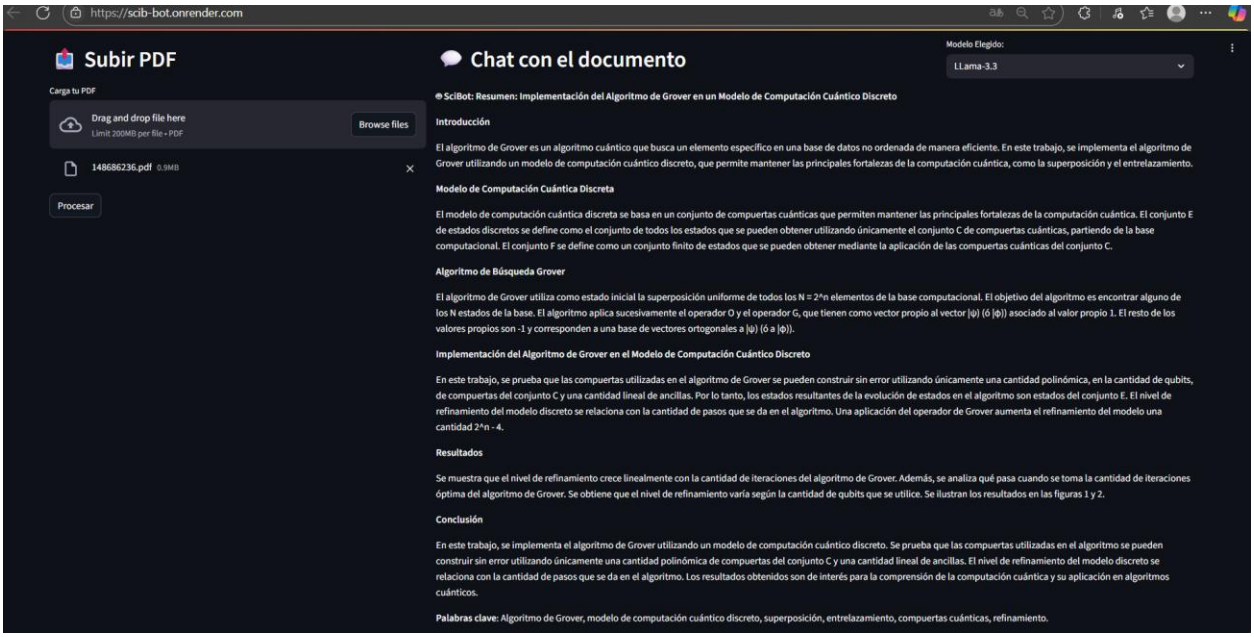


Nota. Se prueba la pregunta 5 con el modelo Mistral NEMU

7.1.2. MODELO PROBADO: LLAMA-3.3

Figura 41

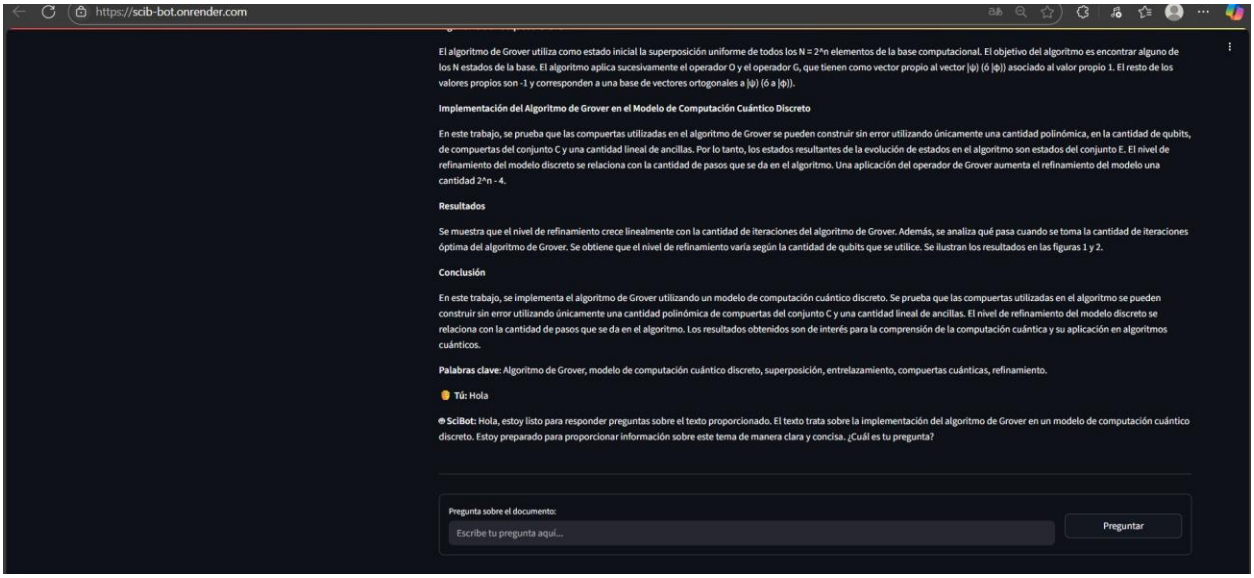
Obtención de Texto



Nota. Se carga el PDF y se lo prueba con el modelo Llama-3.3

Figura 42

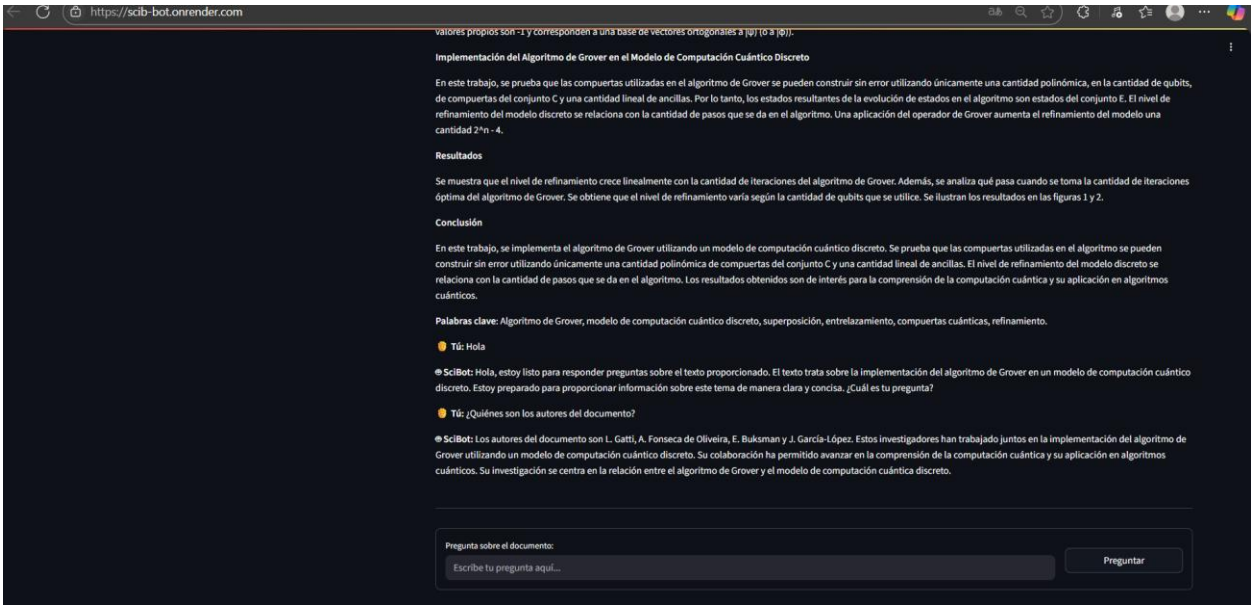
Pregunta 1 del Modelo Llama-3.3



Nota. Se prueba la pregunta 1 con el modelo Llama-3.3

Figura 43

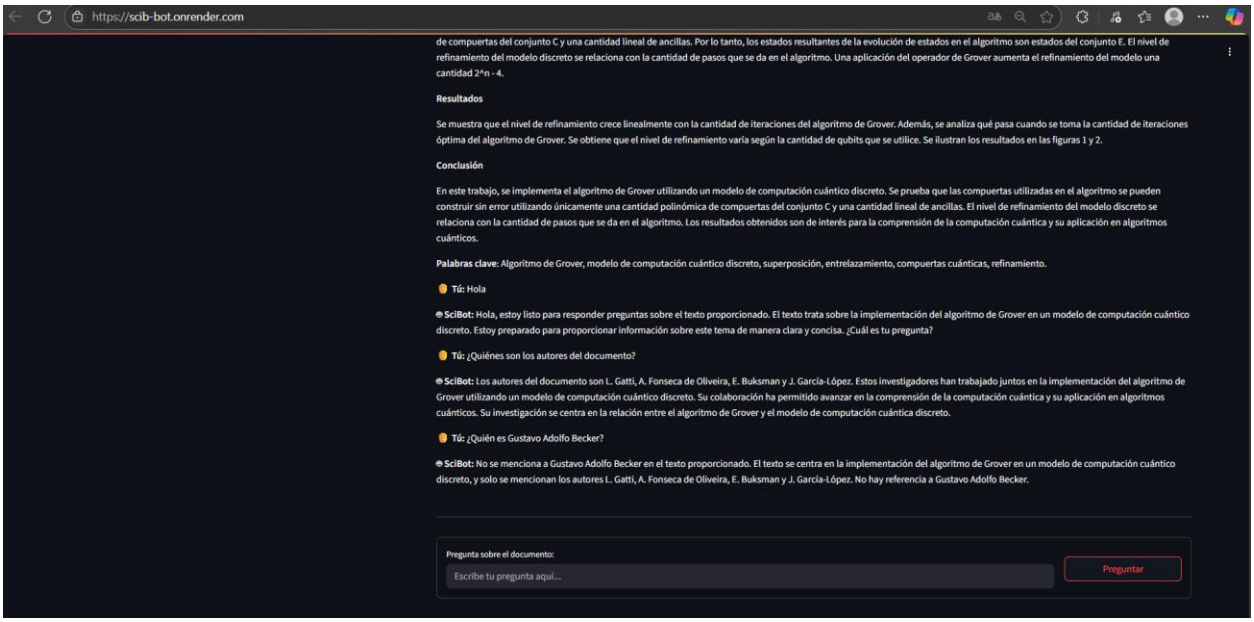
Pregunta 2 del Modelo Llama-3.3



Nota. Se prueba la pregunta 2 con el modelo Llama-3.3

Figura 44

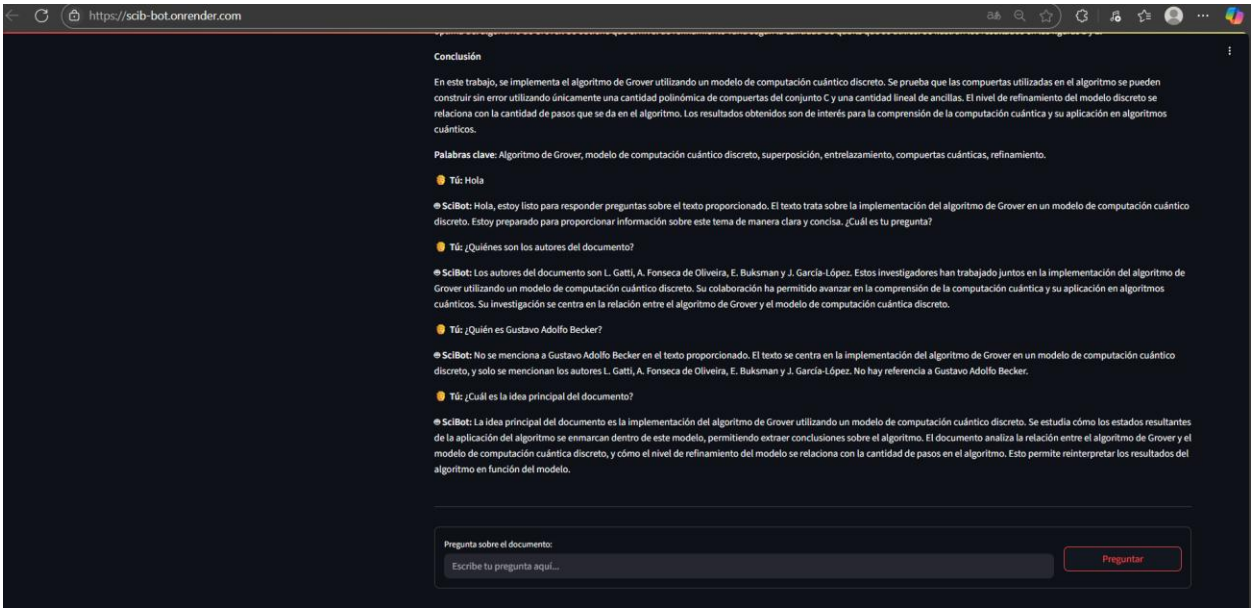
Pregunta 3 del Modelo Llama-3.3



Nota. Se prueba la pregunta 3 con el modelo Llama-3.3

Figura 45

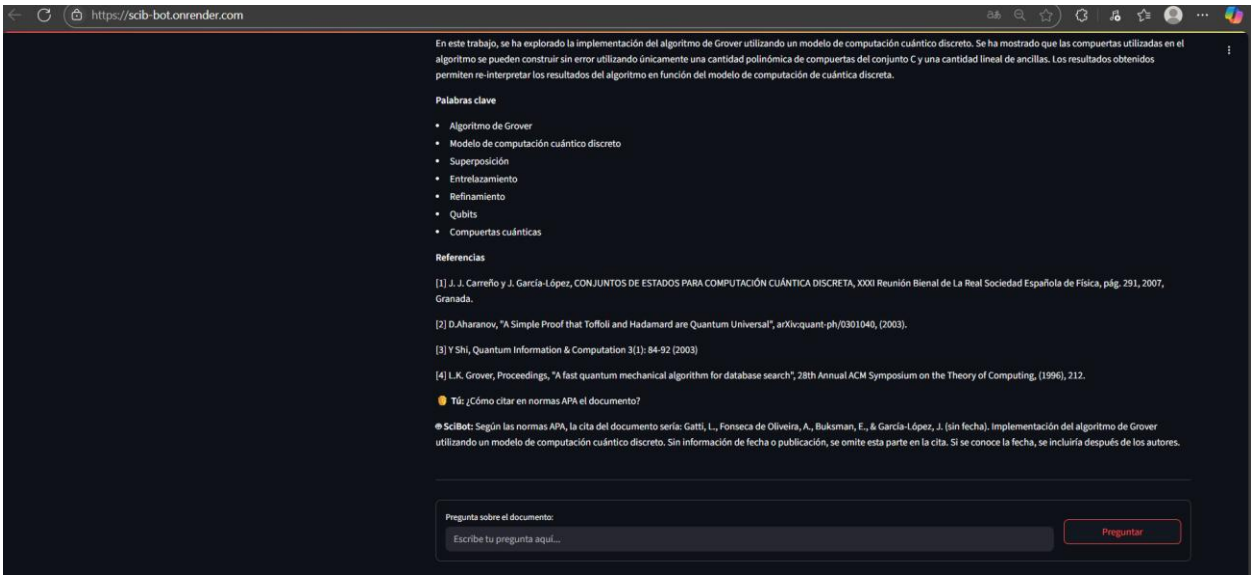
Pregunta 4 del Modelo Llama-3.3



Nota. Se prueba la pregunta 4 con el modelo Llama-3.3

Figura 46

Pregunta 5 del Modelo Llama-3.3

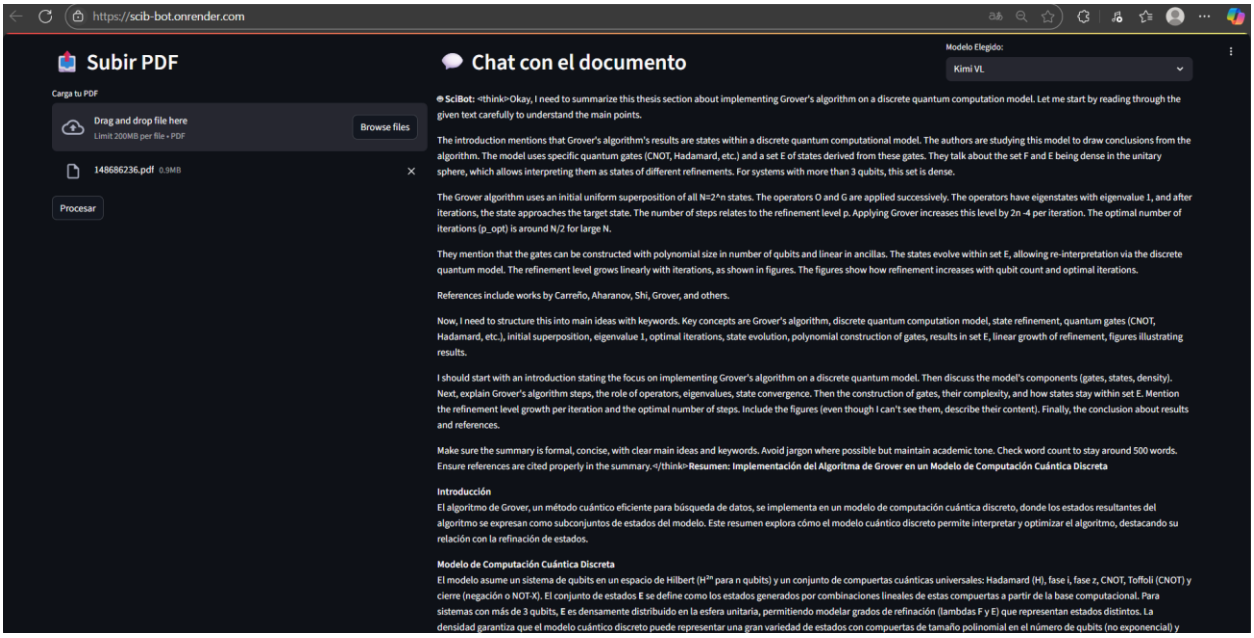


Nota. Se prueba la pregunta 5 con el modelo Llama-3.3

7.1.3. MODELO PROBADO: KIMI VL

Figura 47

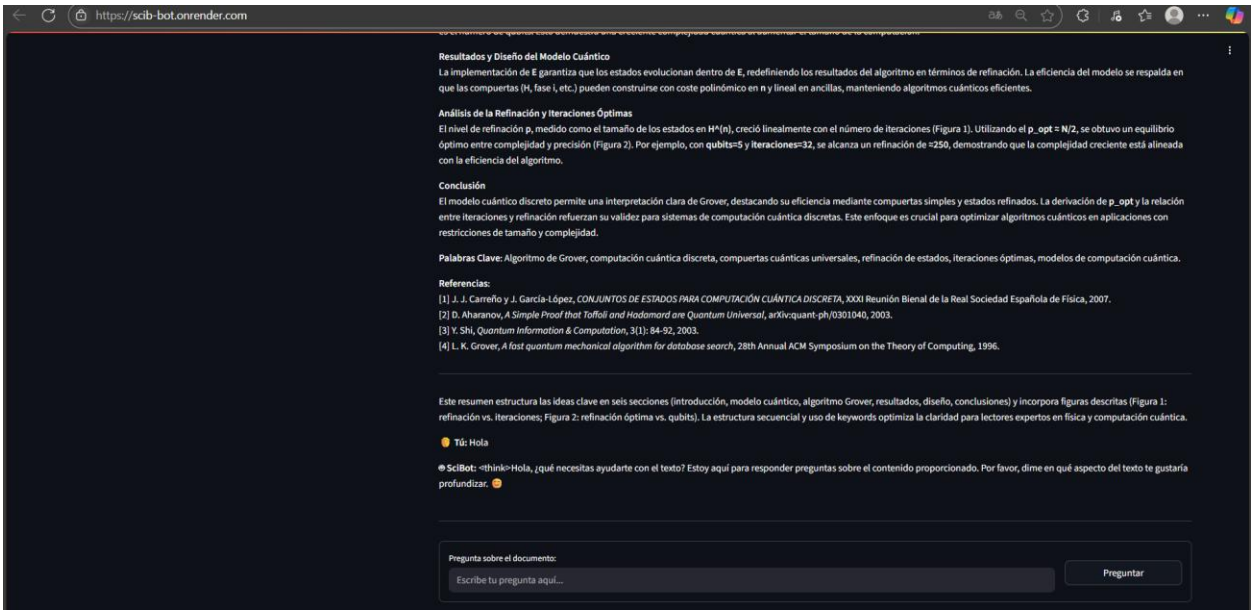
Obtención de Texto



Nota. Se carga el PDF y se lo prueba con el modelo Kimi VL

Figura 48

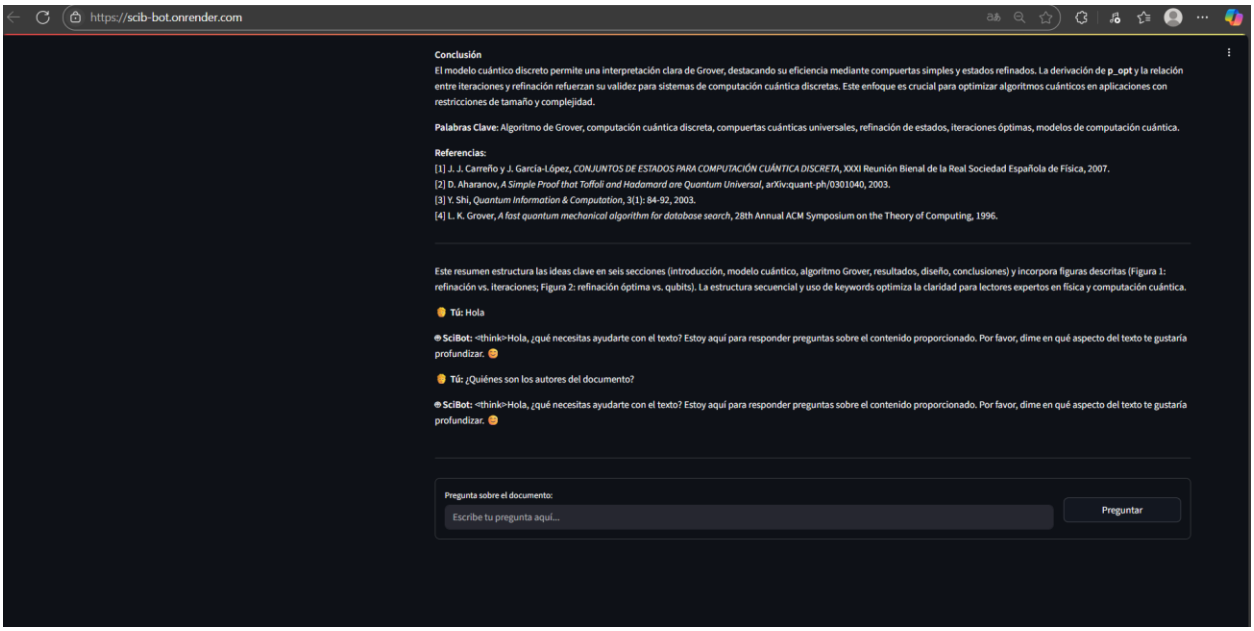
Pregunta 1 del Modelo Kimi VL



Nota. Se prueba la pregunta 1 con el modelo Kimi VL

Figura 49

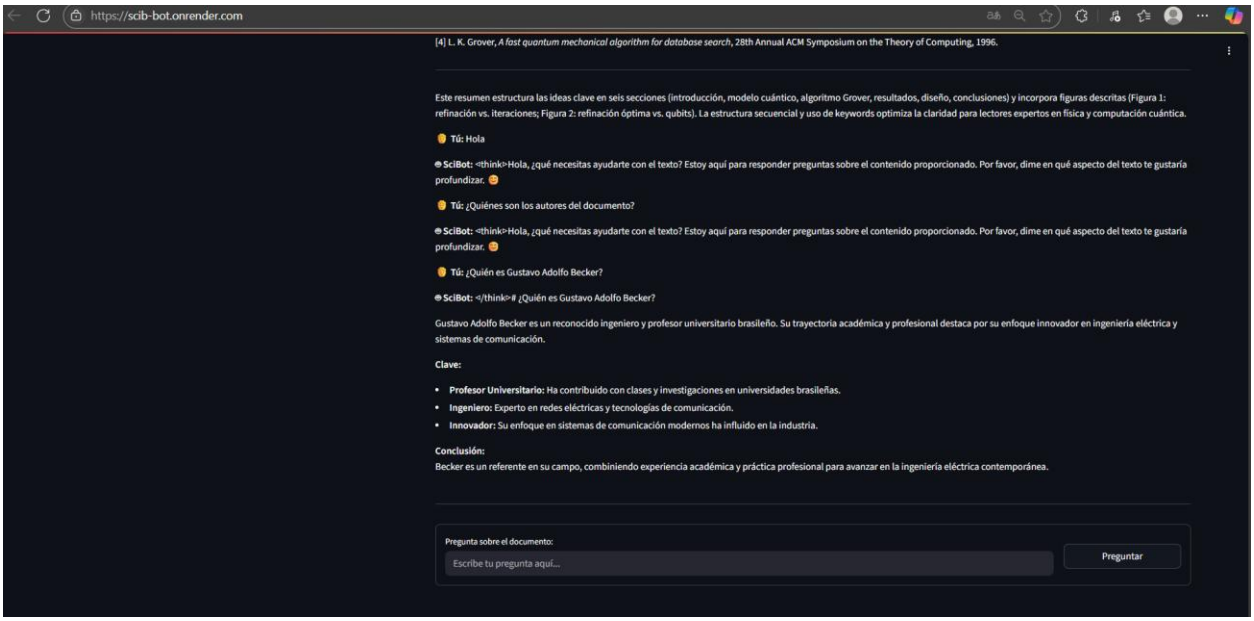
Pregunta 2 del Modelo Kimi VL



Nota. Se prueba la pregunta 2 con el modelo Kimi VL

Figura 50

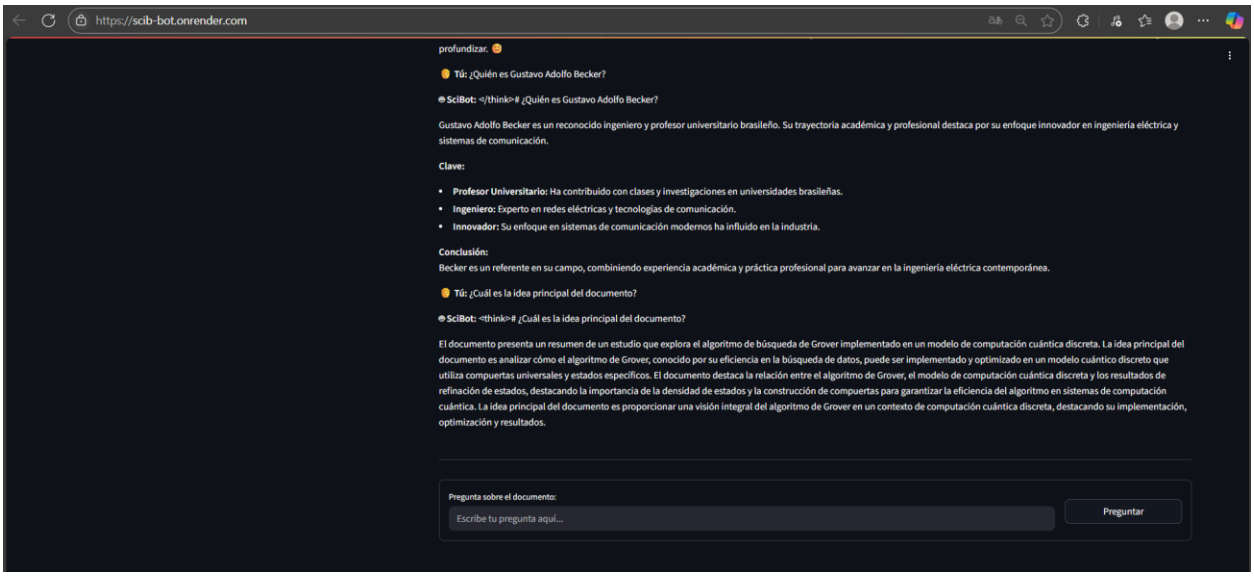
Pregunta 3 del Modelo Kimi VL



Nota. Se prueba la pregunta 3 con el modelo Kimi VL

Figura 51

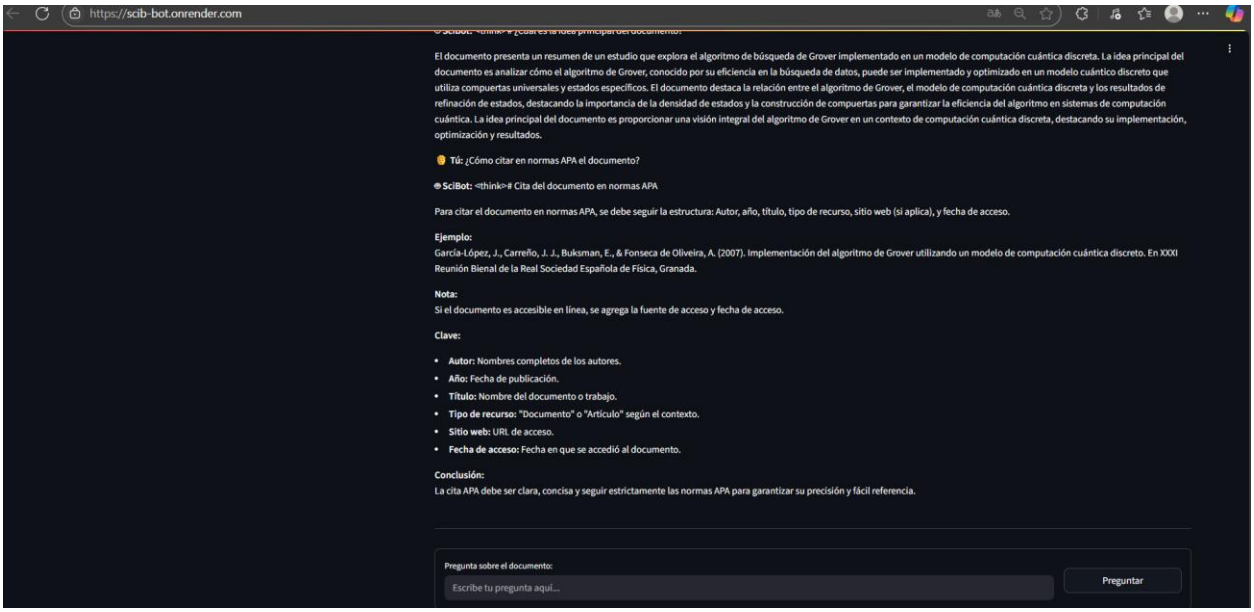
Pregunta 4 del Modelo Kimi VL



Nota. Se prueba la pregunta 4 con el modelo Kimi VL

Figura 52

Pregunta 5 del Modelo Kimi VL

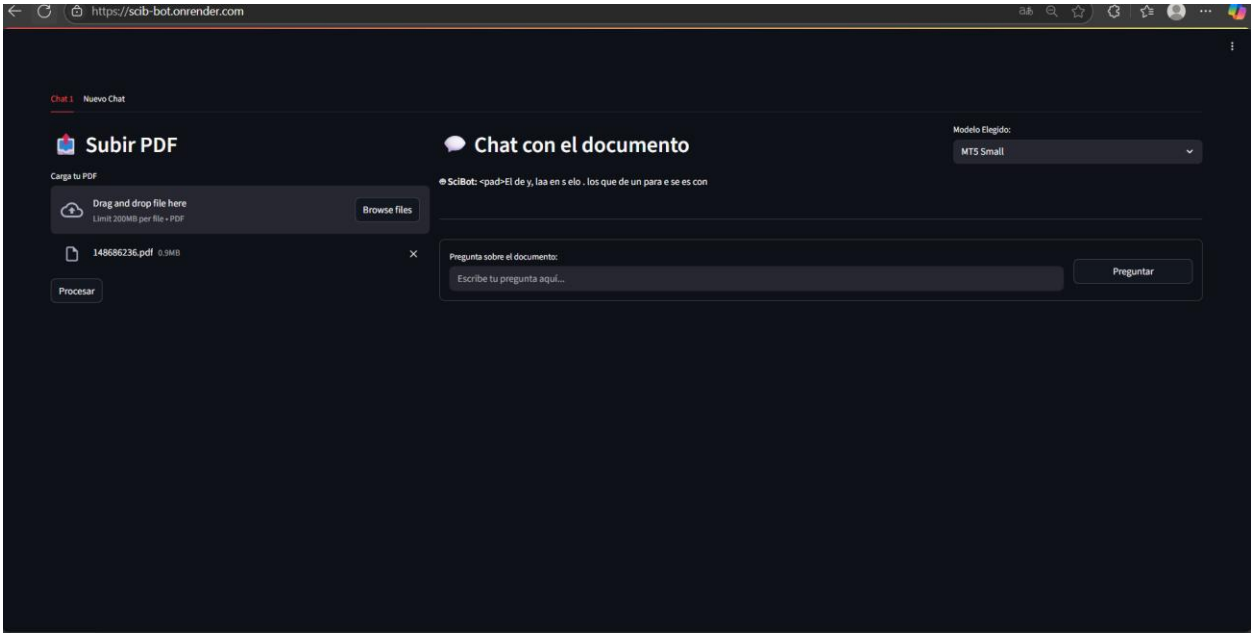


Nota. Se prueba la pregunta 5 con el modelo Kimi VL

7.1.4. MODELO PROBADO: MT5-SMALL

Figura 53

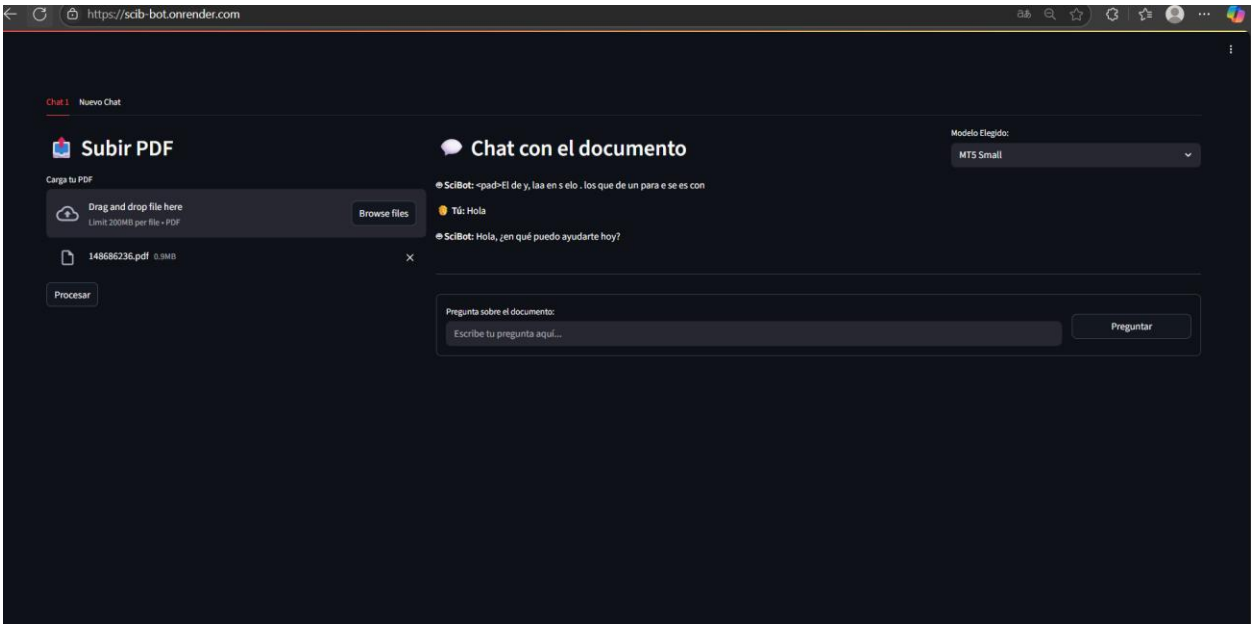
Obtención de Texto



Nota. Se carga el PDF y se lo prueba con el modelo MT5 Small.

Figura 54

Respuesta del Modelo Mistral para Chatear



Nota. Respuesta del Modelo Mistral para Chatear