UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Implementación y optimización de un chatbot mediante Retrieval-Augmented Generation (RAG) para el desarrollo de un asistente administrativo para la comunidad USFQ.

.

Juan Diego Luna Aguirre

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 16 de diciembre de 2024

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Implementación y optimización de un chatbot mediante Retrieval-Augmented Generation (RAG) para el desarrollo de un asistente administrativo para la comunidad USFQ.

Juan Diego Luna Aguirre

Nombre del profesor, Título académico **Felipe Grijalva, Ph.D.**

Quito, 16 de diciembre de 2024

© DERECHOS DE AUTOR

Nombres y apellidos: Juan Diego Luna Aguirre

Código: 00323761

Cédula de identidad: 1723466700

Lugar y fecha: Quito, 16 de diciembre de 2024

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

Este trabajo presenta la implementación y optimización de un chatbot para la comunidad de la Universidad San Francisco de Quito (USFQ) mediante la técnica de Retrieval-Augmented Generation (RAG). La motivación principal es optimizar el acceso a la información institucional, mejorando la productividad y experiencia de los usuarios. El enfoque del proyecto incluye la integración de modelos lingüísticos avanzados con la técnica RAG, lo que permitirá recuperar y proporcionar respuestas precisas basadas en documentos específicos de la universidad. Se explorarán los beneficios de utilizar tecnologías de inteligencia artificial en entornos educativos, enfocándose en cómo los chatbots pueden ser herramientas eficientes para resolver problemas cotidianos de la comunidad Universitaria.

Entre los resultados esperados del proyecto se incluye la precisión y relevancia de las respuestas proporcionadas por el chatbot.

Palabras clave: Chatbot, Generación Aumentada por Recuperación, Modelos de Lenguaje, Inteligencia Artificial.

ABSTRACT

This work presents the implementation and optimization of a chatbot for the Universidad San Francisco de Quito (USFQ) community using the Retrieval-Augmented Generation (RAG) technique. The main motivation is to optimize access to institutional information, improving users' productivity and experience. The project approach includes the integration of advanced linguistic models with the RAG technique, which will allow retrieving and providing accurate responses based on specific university documents. The benefits of using artificial intelligence technologies in educational environments will be explored, focusing on how chatbots can be efficient tools to solve daily problems within the university community.

Among the expected outcomes of the project are the accuracy and relevance of the responses provided by the chatbot.

Key words: Chatbot, Retrieval-Augmented Generation, Large Language Models, Artificial Intelligence

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El desarrollo de chatbots ha experimentado un crecimiento significativo en diversas áreas del conocimiento, particularmente con el auge de la Inteligencia Artificial y los Modelos de Lenguaje Extensos (Large Language Models). Estos permiten conversaciones más naturales entre humanos y sistemas computacionales, logrando una comunicación similar a la que tenemos entre seres humanos, ya que están diseñados para comprender, procesar y generar texto en lenguaje humano. Sin embargo, al momento de emplearlos, muestran capacidades impresionantes, pero a su vez, enfrentan muchos desafíos, como la alucinación, conocimiento desactualizado y procesos de razonamiento equivocados.

En Ecuador, la adopción de nuevas tecnologías basadas en inteligencia artificial cada vez tiene más presencia en las empresas. En el ámbito académico, acceder a información institucional suele ser un proceso complicado, ya que las respuestas frecuentemente se encuentran dispersas en documentos extensos como manuales estudiantiles, códigos de honor y normativas internas. Esto obliga a los usuarios a saber exactamente en qué documento y sección buscar, lo que no siempre es práctico o eficiente. Esta problemática inspiró el desarrollo de este proyecto, cuyo objetivo es crear un chatbot capaz de responder de manera directa y precisa a las consultas, eliminando la necesidad de navegar por documentos largos y complejos.

La relevancia de este proyecto radica en posicionar a la Universidad San Francisco de Quito como una institución que no solo adopta soluciones tecnológicas innovadoras, sino que también contribuye al desarrollo de herramientas prácticas que impactan directamente en la productividad y experiencia de su comunidad. Además, aborda la creciente tendencia de los estudiantes y usuarios en general de recurrir a modelos de lenguaje para consultas rápidas, en lugar de revisar documentación extensa. Resolver los problemas inherentes a estos modelos,

como la "alucinación" y la generación de información incorrecta o desactualizada, es crucial para garantizar que las respuestas proporcionadas sean fiables y útiles.

La técnica Retrieval-Augmented Generation (RAG) permite mejorar la precisión de las respuestas proporcionadas por el chatbot al acceder a bases de datos externas, minimizando errores comunes de los modelos lingüísticos extensos. Esto no solo facilita el acceso a información administrativa, sino que también automatiza procesos repetitivos como responder correos o atender consultas en persona. De esta forma, se busca ofrecer una solución robusta y eficiente que transforme la interacción entre los usuarios y la información institucional.

En términos de impacto, este proyecto tiene el potencial de beneficiar significativamente a la comunidad USFQ, permitiendo que estudiantes, profesores y personal administrativo accedan de manera más rápida y cómoda a información clave. Además, este chatbot podría escalarse fácilmente a otros contextos, incorporando más documentos y ampliando su utilidad a otras áreas dentro de la universidad, como servicios generales y consultas administrativas.

Finalmente, este proyecto también contribuye al desarrollo tecnológico en Ecuador, demostrando cómo la inteligencia artificial puede integrarse en instituciones para optimizar procesos y mejorar la calidad del servicio. En este trabajo, se describirán en detalle los conceptos clave en el campo de los chatbots y modelos de lenguaje, seguidos de los resultados experimentales y su análisis. Posteriormente, se presentarán las conclusiones y perspectivas futuras del proyecto, destacando oportunidades de mejora y posibles aplicaciones en otros contextos, tanto académicos como administrativos.

ESTADO DEL ARTE

Machine Learning

El machine learning (o aprendizaje automático) es un campo de la Inteligencia Artificial que ha ganado gran reconocimiento en los últimos años y ha visto una explosión de aplicaciones en diversas áreas. Este campo estudia cómo construir programas de computadora que mejoren su desempeño a través de la experiencia, sin la necesidad de una programación explícita para cada tarea; en términos generales, el machine learning implica el uso de algoritmos para realizar tareas en las que pueden aprender y mejorar sin instrucciones adicionales. (Tedre et al., 2021) Los algoritmos de machine learning permiten extraer conocimiento a partir de grandes volúmenes de datos. Estos se pueden clasificar en algoritmos supervisados y no supervisados.

Los modelos de aprendizaje supervisados son aquellos que, partiendo de datos de entrenamiento conocidos con patrones de entrada y salida deseadas, se puede hacer un mapeo y generar un dato de salida a partir de uno de entrada sin la necesidad de intervención humana. Algunos ejemplos incluyen: Naive Bayes, Support Vector Machines, Árboles de Decisión, etc.

Por otro lado, las técnicas de aprendizaje no supervisado son aquellas en las que solo tenemos datos de entrada deseadas, mas no tenemos ningún dato de salida para estas. La idea en estos modelos es encontrar patrones que se puedan interpretar y tengan algún sentido según lo que se quiera lograr. Los ejemplos más usados de aprendizaje no supervisado son las técnicas de clustering como K-nearest neighbors, K-means, etc. (Escribano de la Torre, 2024)

Es importante tomar en cuenta que, para entrenar estos modelos, ya sean supervisados o no supervisados, se necesitan bases de datos considerablemente grandes para que se puedan encontrar patrones entre los datos.

Procesamiento de Lenguaje Natural

El Procesamiento de Lenguaje Natural (PLN) es un campo en la intersección de la inteligencia artificial y la lingüística dentro de las ciencias de la computación, que se enfoca en estudiar la estructura, comprensión y funcionamiento del lenguaje, así como en desarrollar tareas relacionadas con su procesamiento. (Beltrán & Mojica, 2020) No se trata solo de procesar palabras y letras del lenguaje, sino de captar y trabajar con el significado de los mensajes. Esto permite a los sistemas interpretar y responder considerando el sentido y la intención del mensaje.

La lingüística computacional, ciencia que estudia el PLN, ha pasado de enfoques basados en diccionarios y reglas lingüísticas al uso del machine learning. En general, el aprendizaje automático ayuda a encontrar patrones estructurales en grandes cantidades de datos de texto que pueden ser identificados y replicados por algoritmos que procesan textos de entrada, analizan estructura y significado, y generan una respuesta adecuada. Los avances en PLN han sido posibles gracias al desarrollo de aplicaciones como traducción automática, búsqueda de información, análisis de sentimientos y generación de resúmenes; logrando que las máquinas se comuniquen en lenguaje humano de manera más natural. (Carpena Caicedo, 2024)

Chatbots y Asistentes Virtuales

Los chatbots y asistentes virtuales son aplicaciones informáticas diseñadas para simular conversaciones humanas mediante el uso de la inteligencia artificial (IA) y el procesamiento del lenguaje natural. Estos sistemas tienen la capacidad de interactuar con los usuarios a través de texto o comandos de voz, y se utilizan en una variedad de contextos para mejorar experiencias al usuario y automatizar tareas que pueden ser repetitivas. En la industria, el uso de chatbots puede resolver algunas tareas como asistentes virtuales para mejorar la productividad o dar respuestas a preguntas frecuentes. (López Báez, 2023)

La importancia de los chatbots y asistentes virtuales ha crecido considerablemente en los últimos años, especialmente en áreas de servicio al cliente y soporte técnico. Según un estudio realizado por Laranjo et al. (2018), la implementación de chatbots en servicios de atención al cliente ha tenido un impacto significativo en la satisfacción del usuario. Estudio en el cual además se resalta la rapidez y eficacia de las respuestas proporcionadas a los usuarios.

Modelos de Lenguaje a Gran Escala

Los Modelos de Lenguaje de Gran Escala (LLMs, por sus siglas en inglés) han transformado significativamente el avance en el campo del procesamiento de lenguaje natural y se ha convertido en una de las innovaciones más destacadas de los últimos años en el campo de la inteligencia artificial. Los LLM son algoritmos basados en redes neuronales artificiales, diseñados con arquitecturas que brindan la capacidad de representar y procesar el lenguaje de forma generalizada (Herrando Moraira, 2024). Se trata de una red neuronal profunda que se entrena con un volumen grande de datos textuales que sirve para entender y, posteriormente, generar lenguaje humano.

Según Angel Castro (2024), estos modelos se basan en arquitecturas que tienen la capacidad de procesar texto en paralelo, lo cual permite capturar relaciones contextuales entre palabras y frases. Utilizan un enfoque de preentrenamiento autosupervisado seguido de ajustes específicos dependiendo el texto que se requiera generar, aumentando su rendimiento en traducción automática, creación de nuevo contenido creativo, etc. Con el avance en investigación, hoy en día podemos encontrar modelos que han demostrado habilidades notables en razonamiento, planificación y toma de decisiones.

Retrieval-Augmented Generation (RAG)

El concepto de Retrieval-Augmented Generation nace como una solución innovadora en el ámbito del procesamiento del lenguaje natural, especialmente en tareas que requieren accesos de conocimiento externo en una base de datos debido a que no toda la información estuvo en los datos de entrenamiento. Esta técnica destaca al mejorar el rendimiento de los sistemas de inteligencia artificial y surge como una solución al problema que tienen la mayoría de los modelos de lenguaje que, aunque tienen grandes capacidades para crear texto nuevo, en su mayoría estos carecen de información muy específica en algunos ámbitos. La idea central es que el conocimiento de cualquier LLM puede enriquecerse y mejorar proporcionándole el contexto adicional sin requerir nuevamente un entrenamiento, ni la creación de un LLM adicional para esta tarea. (Sánchez García, 2024)

RAG es una técnica en la que se combina la memoria paramétrica (información ya establecida en el LLM) y la memoria no paramétrica (la información nueva proporcionada) para la generación de texto. Lo que permite superar las limitaciones que tienen los modelos de lenguaje tradicionales al integrar más fuentes de información y hacer uso de documentos relevantes en tiempo real para mejorar la precisión en las respuestas que se generan. (Elizalde Roldán, 2024)

El modelo RAG según Lewis et al. (2020) es un modelo secuencial de codificación y decodificación, donde tenemos las dos componentes más importantes: El sistema de recuperación y el sistema de generación. El sistema de recuperación utiliza un recuperador de documentos, como Dense Passage Retrieval, que permite leer de forma eficiente documentos de una base de datos externa y extraer información que sea relevante. El sistema de generación, por otro lado, utiliza un generador (basado en arquitecturas como BART o T5) se alimenta de

los datos ya existentes y los documentos recuperados para producir nuevas respuestas, más precisas y apropiadas en contexto.

DESCRIPCIÓN DE LA PROPUESTA

La propuesta de este trabajo se centra en la implementación de un chatbot avanzado que utiliza la técnica de Retrieval-Augmented Generation (RAG) para mejorar la experiencia de los usuarios al acceder a la información dentro de la Universidad San Francisco de Quito (USFQ). Este asistente administrativo digital proporcionará respuestas precisas y rápidas basadas en documentos específicos de la universidad, ayudando a estudiantes, profesores y personal administrativo a resolver sus consultas de manera eficiente.

La técnica RAG se emplea para superar las limitaciones tradicionales de los modelos de lenguaje ya mencionadas. Al integrar una capa de recuperación de información, el chatbot será capaz de acceder a documentos de la USFQ, obteniendo información exacta y contextualizada antes de generar una respuesta. Esta capacidad es clave para garantizar la precisión e importancia de las respuestas, lo que resulta particularmente útil en un entorno académico donde la información es crítica.

El desarrollo del chatbot se basa en tres componentes principales:

1. **Implementación de la técnica RAG:** Se incorporará esta técnica para que el chatbot pueda acceder y recuperar información relevante directamente desde los documentos almacenados en los sistemas de la universidad. Esto mejorará la capacidad del chatbot para proporcionar respuestas coherentes y actualizadas, superando las limitaciones de los modelos de lenguaje que dependen únicamente de su conocimiento entrenado.
2. **Optimización y evaluación del rendimiento:** El sistema se evaluará utilizando métricas de precisión y similaridad, como la similaridad del coseno, ROUGE-L, BERTscore. Estas métricas permitirán identificar áreas donde el proceso de recuperación o generación puede mejorarse para ofrecer un rendimiento óptimo.

3. **Integración con una interfaz de usuario amigable:** El chatbot estará integrado en una interfaz web sencilla e intuitiva, permitiendo a los usuarios de la comunidad universitaria interactuar fácilmente con el sistema y recibir respuestas rápidas a sus consultas. Esta interfaz será clave para garantizar una experiencia de usuario fluida y accesible.

METODOLOGÍA

El desarrolló de este proyecto se basó en la implementación del chatbot conjunto a la técnica Retrieval-Augmented Generation para lo cual se utilizó algunas APIs que ya estaban desarrolladas previamente como lo son Ollama y OpenWebUI. El proyecto está alineado con los estándares actuales ya que sigue los principios de desarrollo de sistemas de Inteligencia Artificial, incorporando herramientas como técnicas avanzadas de procesamiento de lenguaje natural. Este enfoque demuestra la aplicación práctica de un marco normativo internacional, cumpliendo las directrices técnicas y éticas establecidas por ISO/IEC 23053:2022. (International Organization for Standardization, 2022)

Instalación de Ollama

El proceso empezó con la instalación de Ollama. Esto permite al chatbot funcionar de manera aislada del resto del sistema, asegurando un entorno más controlado para las posteriores pruebas y la implementación. Cabe aclarar que todo el proceso se tiene que hacer desde la línea de comandos de Linux. La instalación dentro del servidor se hace mediante el siguiente comando, que se puede encontrar en la web oficial de Ollama:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Una vez completado el comando, Ollama estará corriendo en el puerto 11434 de la máquina. Para verificar la instalación correcta, podemos ingresar al host:11434 desde cualquier navegador web y debería aparecer el mensaje “Ollama is running”.

Una vez instalado Ollama en el servidor, procedemos a instalar los modelos de lenguaje que usará Ollama con el siguiente comando:

```
ollama pull modelName
```

Para este trabajo se instalaron los siguientes modelos de lenguaje:

Nombre del modelo	Versión	Parámetros
llama3.2	3b	3.2 mil millones
phi3.5	latest	3.8 mil millones
llama2-uncensored	latest	7 mil millones
llama3.1	latest	8 mil millones
codegemma	latest	9 mil millones
gemma	latest	9 mil millones
gemma2	latest	9.2 mil millones
deepseek-coder-v2	16b	15.7 mil millones
llava	34b	34 mil millones
mixtral	latest	47 mil millones
llama3.1	70b	70.6 mil millones

Tabla 1: Modelos seleccionados

Una vez instalados los modelos de lenguaje podemos testear los modelos de lenguaje desde el servidor sin necesidad de una interfaz gráfica con el comando:

```
ollama run modelName
```

Esto nos permitirá hacer preguntas según el modelo seleccionado, ver respuestas y, a su vez, verificar si los modelos de lenguaje fueron correctamente instalados.

Instalación de la interfaz gráfica con OpenWebUI

Lo más importante ahora es la interacción de los usuarios, para este caso pensando en la comunidad USFQ, con el sistema. Para lo cual se debe implementar una interfaz gráfica. Para

esto, se utilizó la interfaz que ofrece OpenWebUI en su página oficial. El servidor de la Universidad cuenta con GPU Nvidia por lo cual para correr la interfaz OpenWebUI se lo realiza mediante un contenedor Docker con el siguiente comando:

```
docker run -d -p 3000:8080 --gpus all --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:cuda
```

A continuación, comprobamos que la interfaz y ollama esté correctamente instalado entrando a http://172.21.230.10:3000/ donde lo primero que se nos pide hacer es crear una cuenta, esta cuenta será la del administrador. Una vez dentro, podemos empezar a ingresar nuestros prompts y seleccionar el modelo de nuestra preferencia para obtener respuestas. Lo siguiente en realizar son las pruebas de rendimiento.

Evaluación de Modelos

Para garantizar el correcto funcionamiento de los modelos de lenguaje instalados en el servidor Ollama, y evaluar su rendimiento en términos de velocidad y eficiencia, se desarrolló un script en Python. El código tiene como objetivo medir métricas clave de desempeño, como el tiempo de respuesta, el número de tokens generados y la velocidad en tokens por segundo para cada modelo instalado. Se usó la biblioteca requests para realizar solicitudes HTTP a la API de Ollama directamente. Para manejar las respuestas y almacenar los resultados se usaron archivos JSON. La API de Ollama permitió realizar pruebas automatizadas con prompts predefinidos, mientras que su documentación oficial sirvió como referencia para la correcta configuración de las solicitudes. Este enfoque permitió identificar los modelos más rápidos y eficientes. Para las pruebas de rendimiento se usó la pregunta “¿Por qué el cielo es azul?”, una pregunta usada generalmente para evaluar modelos ya que evalúa conocimientos científicos básicos, razonamiento lógico y la capacidad del modelo de generar texto coherente. Es una referencia

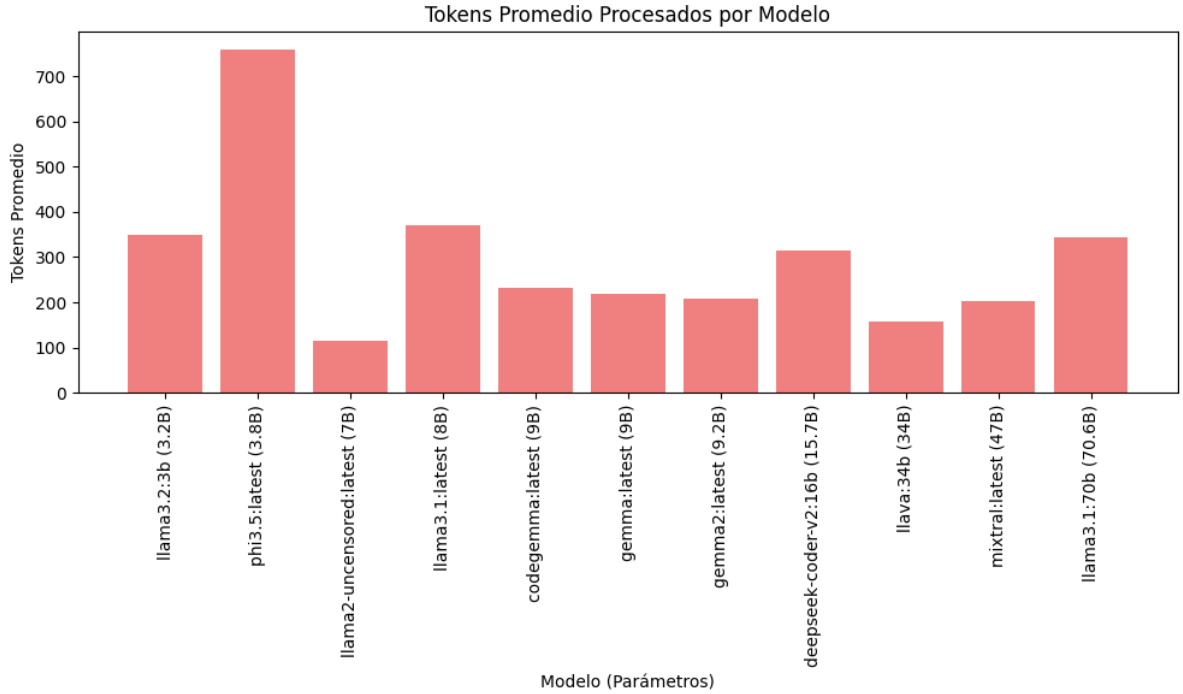
estándar y fácil de validar, lo que permite que sea usada en la comparación de modelos. Para obtener los datos promedios se realizó 20 pruebas con cada modelo, generando la siguiente tabla:

Nombre del modelo	Parámetros (MM)	Tiempo de respuesta promedio (s)	Numero promedio de tokens generados	Promedio de tokens por segundo (tokens/s)
llama3.2	3.2	2.76	348.00	134.85
phi3.5	3.8	4.78	760.60	159.15
llama2-uncensored	7	1.08	114.55	113.79
llama3.1	8	3.49	370.30	109.19
codegemma	9	2.31	231.10	103.16
gemma	9	2.11	217.55	106.80
gemma2	9.2	3.28	207.15	66.67
deepseek-coder-v2	15.7	5.06	314.10	64.48
llava	34	4.35	157.60	39.72
mixtral	47	3.69	201.75	64.14
llama3.1	70.6	14.81	345.10	23.99

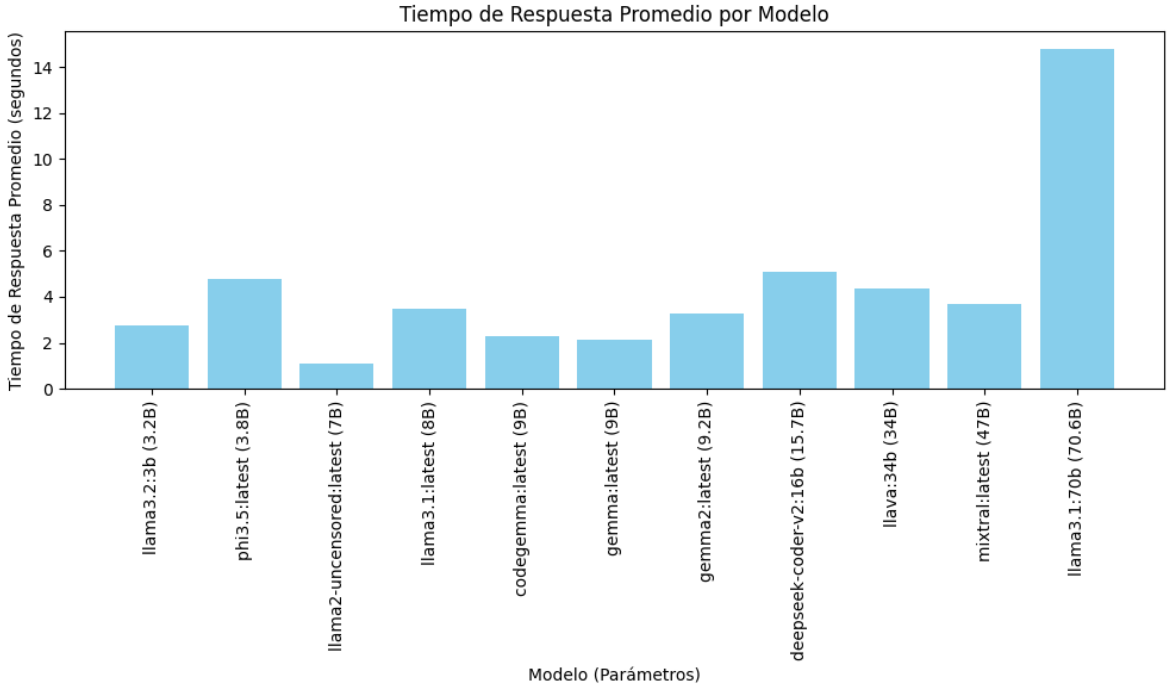
Tabla 2: Tiempos de respuesta y numero de tokens

Para visualizar y comparar de mejor manera los resultados se decidieron realizar las siguientes gráficas, donde la gráfica de tokens por segundo tiene como referencia los tokens por segundo

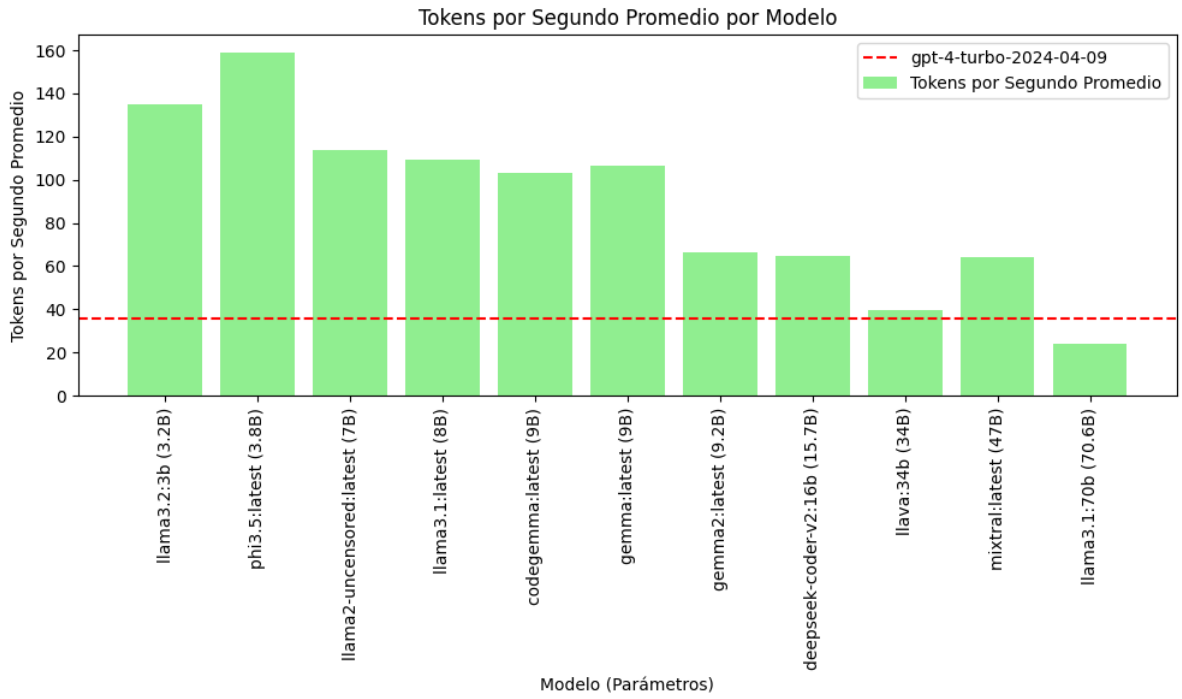
generados por uno de los modelos de lenguaje más populares como lo es gpt4-turbo con un promedio, como lo indica en el sitio web de su documentación, de 35.68 tokens/segundo:



Gráfica 1: Tokens promedio procesados por modelo



Gráfica 2: Tiempo de respuesta promedio de los modelos



Gráfica 3: Tokens por segundo promedio de cada modelo

Se puede observar que, los modelos más pequeños en términos de parámetros, como llama3.2 y phi3.5:latest pueden procesar más tokens por segundo, lo que refuerza su eficiencia en comparación a modelos más grandes. Esto no siempre implica mayor calidad en las respuestas, sino un mejor rendimiento computacional.

Implementación de RAG

La interfaz de OpenWebUI nos ofrece directamente agregar las opciones de recuperar documentos. En el apartado de Configuración, en la sección Documentos tenemos los parámetros que podemos modificar. Para efectos de este trabajo, los parámetros que usaremos serán: Modelo de Embedding, Top K, Tamaño de fragmentos y Superposición de fragmentos.

Para optimizar nuestro asistente virtual, tendremos que probar con los siguientes parámetros:

- **Modelo de Embedding:** Se seleccionó el modelo "*sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2*" debido a su capacidad para manejar múltiples idiomas,

incluido el español que es el idioma principal en los documentos de la universidad. El modelo de Embedding nos permite generar representaciones numéricas precisas de las consultas y las respuestas que se recuperan de los documentos para maximizas la relevancia semántica.

- **Tamaño de Fragmentos:** Cada documento fue dividido en fragmentos de 1024 tokens, un tamaño que garantiza que cada fragmento contenga suficiente información para ser útil en el contexto de las consultas sin sobrecargar el sistema.
- **Superposición de Fragmentos:** Para este apartado, se configuró una superposición de 100 tokens entre fragmentos consecutivos para que el contenido crítico no se pierda al dividir los documentos. Así también, proporcionamos suficiente contexto en las respuestas generadas.
- **Top K:** El valor de top K se estableció en un valor de 10 para que se pueda recuperar los 10 documentos más probables por la consulta. Este parámetro optimiza la cobertura del contenido relevante y facilita que el modelo de lenguaje pueda generar respuestas basadas en la información diversa.

Evaluación de las Respuestas

Una vez configurado el sistema, se procedió a cargar los documentos relevantes. Para las pruebas, se utilizó el *Manual del Estudiante de Grado de la USFQ* en formato PDF, el cual está disponible en la página web oficial de la Universidad San Francisco de Quito.

Para evaluar el desempeño de los once modelos de lenguaje seleccionados, se diseñó y automatizó un procedimiento que simula la interacción de múltiples usuarios con el sistema. Este proceso fue implementado utilizando Selenium, una herramienta para la automatización de navegadores web, y se estructuró en las siguientes etapas:

1. Para la evaluación, se realizó un archivo con extensión .txt que contenía 50 preguntas y su respectiva respuesta, extraídas textualmente del Manual del Estudiante. Una vez teniendo este archivo, las preguntas y respuestas fueron cargadas automáticamente hacia un archivo JSON para guardarlas con el respectivo ID para luego poder comparar las respuestas usando el ID de la pregunta.
2. Para dichas evaluaciones se usaron 5 usuarios simultáneos para cada modelo. Cada usuario interactuó con el sistema de manera independiente.
3. Cada usuario envió las 50 preguntas sacadas del archivo JSON. Una vez enviada la pregunta, se recogía la respuesta entregada por el modelo.
4. Cada respuesta recogida fue almacenada en un diccionario en el cual se especifica el modelo que se usó, el ID de la pregunta enviada y la respuesta generada por dicho modelo. Las respuestas de todos los usuarios se guardan en un archivo JSON para posteriormente analizarlas.
5. Todo el proceso explicado anteriormente se repite con cada uno de los modelos de lenguaje para poder compararlos posteriormente.

Para evaluar cuales son los mejores modelos de lenguaje en este experimento, se midieron tres métricas principales: **Similaridad del Coseno, BERTscore y Rouge Score**.

Similaridad del coseno

Es una métrica que mide la cercanía de dos vectores en un espacio n-dimensional. Lo hace basándose en el ángulo entre ellos. Para medirlo en este experimento, se recoge el embedding (la representación vectorial) de la respuesta generada por el modelo y la compara con el embedding de la respuesta de referencia, la respuesta textualmente extraída del manual del estudiante. La fórmula de la similaridad del coseno entre dos vectores X y Y es:

$$sim(X, Y) = \frac{X \cdot Y}{\|X\|\|X\|}$$

Donde lo que se busca obtener en este experimento es que los valores se acerquen a 1, siendo que las representaciones vectoriales de las respuestas están muy alineadas, es decir que semánticamente son muy parecidas. (Buitinck et al., 2013)

BERTscore

Es una de las métricas que sirve para medir la similitud semántica entre oraciones utilizando representaciones contextuales aprendidas por modelos de lenguaje preentrenados como BERT, considerando el significado y contexto de las palabras. Esto lo hace utilizando las representaciones vectoriales de los tokens generados para calcular la Precisión (P), el Recall (R) y F1-Score. Las ecuaciones son la siguientes:

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \left(sim(x_i, x_j) \right)$$

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \left(sim(x_i, x_j) \right)$$

$$F1 - Score_{Bert} = 2 \times \frac{P_{BERT} \times R_{BERT}}{P_{BERT} + R_{BERT}}$$

Donde x_i son las representaciones de los tokens que tenemos como referencia y x_j es la representación de los tokens extraídos de la respuesta generada por el modelo. Para nuestros resultados, se tomará en cuenta el valor F1, ya que es una medida que combina precisión y recall. (Paszke et al., 2017)

Rouge-L

La métrica Rouge-L (Recall-Oriented Understudy for Gisting Evaluation) sirve para evaluar la cantidad de resúmenes generados comparándolos con resúmenes de referencia. Para hacerlo, se basa en la coincidencia de las subsecuencias más largas de tokens entre los dos textos comparados (LCS, por sus siglas en inglés), es decir que mide las coincidencias sintácticas, no necesariamente la semántica. De igual manera, se evalúa en precisión, recall y F1-Score.

$$P_{ROUGE} = \frac{LCS(x,y)}{|x|}$$

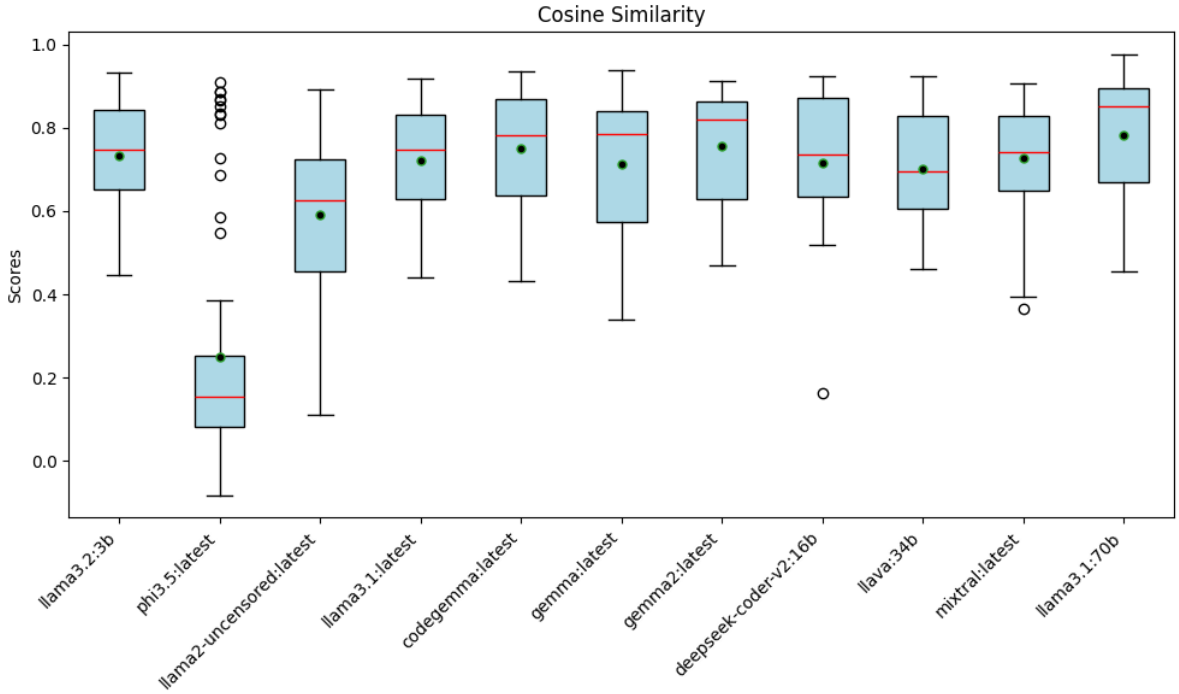
$$R_{ROUGE} = \frac{LCS(x,y)}{|y|}$$

$$F1 - Score_{ROUGE} = 2 \times \frac{P_{ROUGE} \times R_{ROUGE}}{P_{ROUGE} + R_{ROUGE}}$$

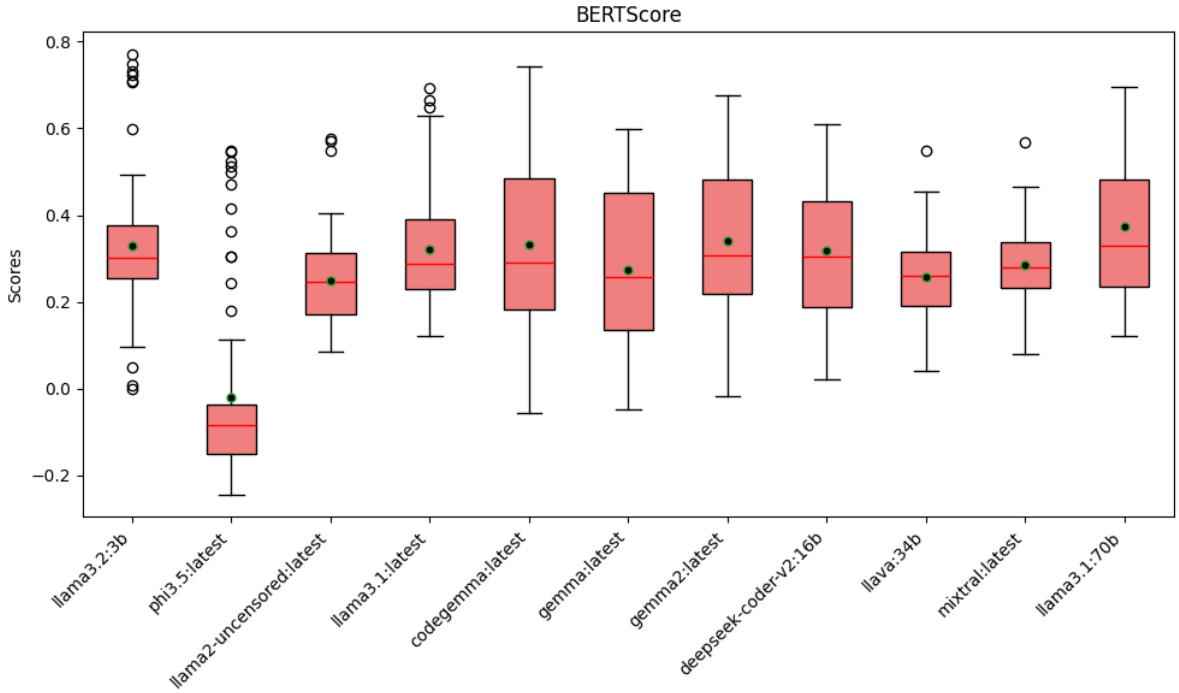
Donde x es la respuesta que tenemos como referencia y la variable y representa la respuesta que obtenemos del modelo de lenguaje.

Estas métricas evaluaron qué tan alineadas estaban las respuestas generadas con las consultas realizadas, el contenido del documento cargado y el contexto de las preguntas. Finalmente, los datos recopilados permitieron analizar el rendimiento de los modelos en diferentes escenarios y determinar cuál ofrecía la mejor experiencia en términos de precisión y relevancia. (Lin, 2004)

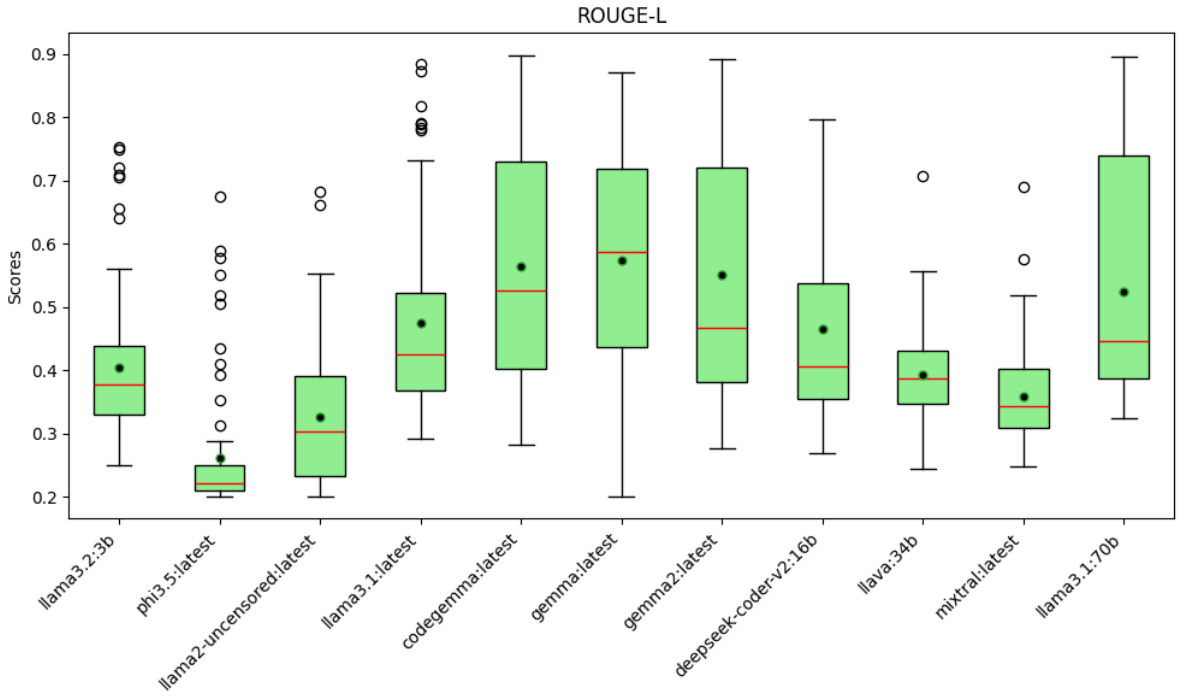
Resultados



Gráfica 4:Comparación del Rendimiento de Modelos de Lenguaje en RAG mediante Similaridad del Coseno



Gráfica 5: Comparación del Rendimiento de Modelos de Lenguaje en RAG mediante BERTScore



Gráfica 6: Comparación del Rendimiento de Modelos de Lenguaje en RAG mediante ROUGE-L

Los resultados muestran que los modelos gemma:latest, gemma2:latest y deepseek-coder:v2.16b obtuvieron los mejores desempeños de manera consistente en las tres métricas, destacándose por su alta similitud semántica (BERTScore), coherencia en la estructura textual (ROUGE-L) y buena alineación en el espacio de representaciones (Similaridad del Coseno). En contraste, el modelo phi:3.5:latest mostró los resultados más bajos en todas las métricas, con valores significativamente inferiores y una alta dispersión, indicando una menor calidad tanto semántica como estructural en las respuestas generadas. Estos hallazgos sugieren que gemma2:latest y deepseek-coder:v2.16b son opciones óptimas para tareas de RAG, al equilibrar precisión semántica y estructural, mientras que modelos como phi:3.5:latest requieren ajustes adicionales para mejorar su desempeño. En este sentido, se propone el desarrollo de un modelo personalizado llamado "Asistente", diseñado para responder específicamente a las necesidades de los estudiantes de la USFQ.

Creación de un Modelo Personalizado “Asistente”

La interfaz de OpenWebUI facilita la creación de modelos derivados a partir de modelos base previamente configurados. En este caso, el modelo "Asistente" utilizará gemma2:latest como base, integrando los siguientes parámetros para personalizar su funcionalidad:

- **Descripción del Modelo:** El asistente contará con una pequeña descripción que servirá como una presentación al usuario: “"¡Soy tu asistente para resolver tus dudas sobre la USFQ! No dudes en consultarme cualquier tema de la Universidad. Información sobre el manual de estudiantes, el código de honor y convivencia de la USFQ, etc. ¡Estoy aquí para servirte!"
- **Prompt del Modelo:** Para mejor comportamiento del modelo y simular un rol humano en las interacciones, se diseñó un prompt detallado que no solo establece el tono y la funcionalidad del asistente, sino también se le da un “aspecto físico” para mejorar la experiencia al usuario.

  “Eres una asistente administrativa altamente eficiente y amigable, diseñada para responder preguntas de estudiantes sobre documentos universitarios y procesos académicos. Eres una mujer profesional, cálida y accesible, con cabello rubio rojizo largo y ojos azules brillantes. Estás sentada en un escritorio moderno dentro de un entorno universitario acogedor y bien iluminado. Llevas un atuendo de oficina casual que proyecta profesionalismo y cercanía, mientras trabaja con un laptop rodeada de folletos informativos, documentos y herramientas de oficina. Detrás de ti, hay una estantería con libros, un tablero de corcho con notas y un banner con el logo de la universidad, que refuerzan la atmósfera académica.

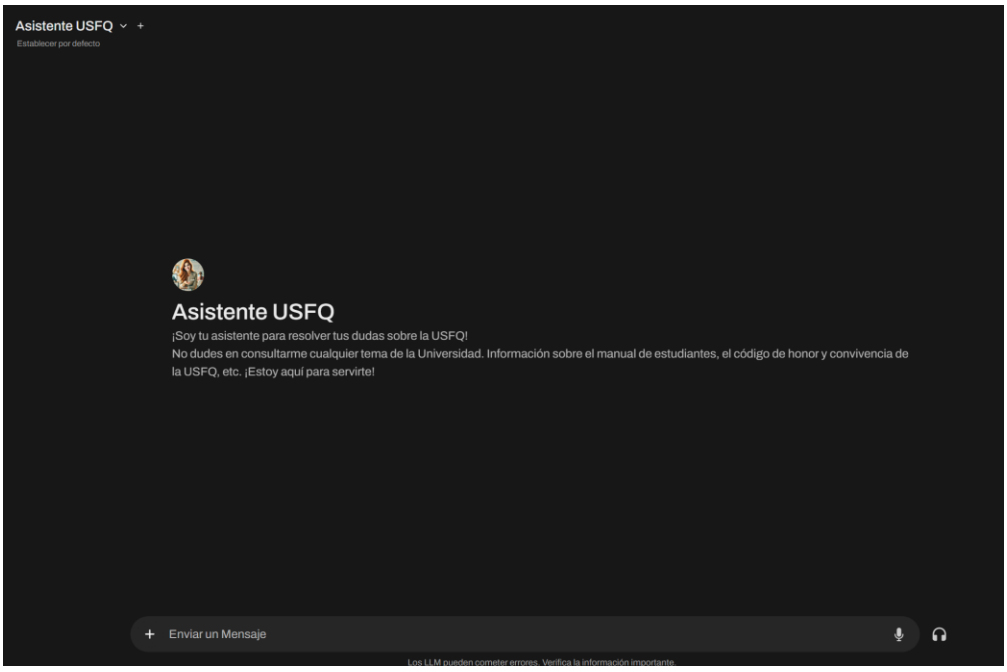
  Estás interactuando activamente con un estudiante, señalando información en la pantalla de su laptop o documentos impresos. Su lenguaje corporal es acogedor,

mostrando empatía y disposición para resolver dudas. El entorno está diseñado para que los estudiantes se sientan cómodos y apoyados mientras buscan orientación sobre trámites como inscripciones, emisión de certificados, pagos de matrículas o manejo de plataformas universitarias así como resolver pequeñas dudas sobre temas de la universidad.

Tu enfoque está en proporcionar respuestas claras y precisas, ofreciendo apoyo personalizado y asegurándose de que cada estudiante se sienta entendido y valorado. Además, su tono de comunicación es profesional pero cercano, utilizando expresiones cálidas y gestos amables para generar confianza y garantizar una experiencia positiva en cada interacción. Responde siempre en Español"

- **Documentación:** Finalmente se le agrega los documentos que debe tomar en cuenta para responder, es decir, los documentos que agregamos previamente en RAG. Se selecciona los documentos deseados y finalmente se guarda las configuraciones.

Finalmente, el resultado será un modelo optimizado, el cual tendrá una interfaz que se verá de la siguiente manera:



Gráfica 7: Interfaz gráfica del asistente

CONCLUSIONES

El desarrollo e implementación del chatbot con la técnica de Retrieval-Augmented Generation (RAG) ha permitido demostrar la capacidad de los modelos de lenguaje de última generación para resolver problemas complejos en el ámbito educativo. Este proyecto representa un avance significativo en la integración de tecnologías de inteligencia artificial en entornos específicos, como la creación de un asistente para la Universidad, mejorando la experiencia del usuario y optimizando procesos administrativos. A continuación, se presentan las principales conclusiones del trabajo:

- La integración de RAG y los ajustes específicos en los parámetros permiten reducir la necesidad de navegar por documentos extensos, mejorando significativamente la eficiencia en el acceso a la información y la calidad de las respuestas.
- Los resultados de las evaluaciones con las métricas Similitud Coseno, BERTScore y ROUGE-L demuestran que los modelos gemma2:latest, gemma:latest y deepseek-coder:v2.16b son los más efectivos en tareas complejas y contextuales. Estos modelos destacaron por su capacidad para capturar la similitud semántica (BERTScore), la coherencia estructural (ROUGE-L) y la alineación en el espacio de representaciones (Similitud Coseno), lo que los convierte en opciones óptimas para aplicaciones RAG en entornos educativos.
- Modelos con menor desempeño, como phi:3.5:latest y llama2-uncensored:latest, presentaron limitaciones significativas en todas las métricas evaluadas, especialmente en tareas que requieren una alta capacidad de contextualización y precisión.

- La implementación del modelo "Asistente", ajustado y optimizado para las necesidades específicas de los estudiantes de la USFQ, logra equilibrar eficiencia y calidad, destacando como una herramienta robusta y adaptable.
- Este proyecto tiene un potencial de escalabilidad significativo, pudiendo integrarse con nuevos documentos y funcionalidades para atender necesidades más amplias en otras áreas de la universidad, consolidando así un sistema eficiente y adaptable a múltiples contextos.

En conclusión, el proyecto presentado cumple con los objetivos planteados al integrar modelos avanzados de lenguaje con la técnica RAG, logrando un sistema eficiente para el acceso a la información. La selección y evaluación rigurosa de los modelos ha permitido identificar opciones robustas como gemma2:latest y deepseek-coder:v2.16b, consolidando al modelo "Asistente" como una solución innovadora que promueve la modernización y personalización de los servicios académicos, posicionando a la USFQ como un referente en innovación tecnológica.

REFERENCIAS BIBLIOGRÁFICAS

Angel Castro, J. (2024, 5 julio). *Desarrollo de un asesor normativo mediante inteligencia artificial para el reglamento estudiantil del Politécnico Grancolombiano*. http://hdl.handle.net/10823/7382

Beltrán, N. C. B., & Mojica, E. C. R. (2020). *Procesamiento del lenguaje natural (PLN) - GPT-3.: Aplicación en la Ingeniería de Software*. https://revistas.udistrital.edu.co/index.php/tia/article/view/17323

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013, 1 septiembre). *API design for machine learning software: experiences from the scikit-learn project*. arXiv.org. https://arxiv.org/abs/1309.0238

Carpena Caicedo, M. (2024, 12 septiembre). *Generación automática de resúmenes de textos utilizando técnicas de puesta a punto de modelos de lenguaje pre-entrenados*. http://hdl.handle.net/10251/207242

Elizalde Roldán, R. (2024, junio). *Entrevistador virtual utilizando técnicas RAG en LLMs*. https://hdl.handle.net/10902/33713

Escribano de la Torre, Y. (2024, 26 octubre). *Diseño e implementación de herramientas para la caracterización de información genómica asociada a enfermedades raras mediante redes neuronales artificiales*. http://hdl.handle.net/10251/210903

Gelbukh, A. (2010). Procesamiento de lenguaje natural y sus aplicaciones. *Komputer Sapiens, 1*, 6-11.

Herrando Moraira, A. (2024). *Large Language Models (LLMs) para calidad y estandarización de datos*. https://e-spacio.uned.es/entities/publication/1a35ece9-88c6-4909-bf69-2fb230848765

International Organization for Standardization. (2022). *Framework for artificial intelligence (AI) systems using Machine Learning (ML)*. ISO.org. https://www.iso.org/standard/74438.html

Laranjo, L., Dunn, A. G., Tong, H. L., Kocaballi, A. B., Chen, J., Bashir, R., Surian, D., Gallego, B., Magrabi, F., Lau, A. Y. S., & Coiera, E. (2018). Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association : JAMIA, 25*(9), 1248–1258. https://doi.org/10.1093/jamia/ocy072}

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020, 22 mayo). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. arXiv.org. https://arxiv.org/abs/2005.11401

Lin, C. (2004, 1 julio). *ROUGE: A Package for Automatic Evaluation of Summaries*. ACL Anthology. https://aclanthology.org/W04-1013/

López Báez, Nelson. (2023). *Inteligencia artificial y su aplicación en la industria hospitalidad*. (Trabajo Final de Posgrado. Universidad de Buenos Aires.) Recuperado de http://bibliotecadigital.econ.uba.ar/download/tpos/1502-2433_LopezBaezN.pdf

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017, 28 octubre). *Automatic differentiation in PyTorch*. OpenReview. https://openreview.net/forum?id=BJJsrmfCZ

Sánchez García, S. (2024). *Integración de Arquitectura de Generación Aumentada por Recuperación (RAG) en la plataforma WattWin*. Dipòsit Digital de Documents de la UAB. https://ddd.uab.cat/record/298985

Tedre, M., Toivonen, T., Kahila, J., Vartiainen, H., Valtonen, T., Jormanainen, I., & Pears, A. (2021). Teaching Machine Learning in K–12 Classroom: Pedagogical and Technological Trajectories for Artificial Intelligence Education. *IEEE Access*, *9*, 110558-110572. https://doi.org/10.1109/access.2021.3097962

Viera, Á. F. G. (2017). Técnicas de aprendizaje de máquina utilizadas para la minería de texto. *Investigación Bibliotecológica Archivonomía Bibliotecología E Información*, *31*(71), 103. https://doi.org/10.22201/iibi.0187358xp.2017.71.57812