

Implementación de un Sistema RAG para el Curso de Introducción a la Programación de la Universidad de Matanzas

MSc. Yosbel Peñate Barceló

January 1, 2025

Contents

1	Introducción	3
1.1	Contexto y Motivación	3
1.2	Objetivos del Proyecto	3
1.3	Alcance del Proyecto	4
2	Diseño del Sistema RAG	5
2.1	Arquitectura General del Sistema	5
2.2	Preprocesamiento de los Documentos	7
2.3	Modelo de Recuperación	7
2.4	Modelo de Generación	7
2.5	Integración del Modelo de Recuperación y Generación	8
3	Implementación del Sistema	8
3.1	Herramientas y Tecnologías Utilizadas	8
3.2	Desarrollo del Pipeline de RAG	8
3.3	Despliegue del Sistema (si aplica)	8
4	Evaluación del Sistema	8
4.1	Diseño de la Evaluación	8
4.2	Resultados de la Evaluación	9
5	Análisis y Discusión	9
5.1	Beneficios del Sistema RAG	9
5.2	Desafíos y Limitaciones	9
6	Conclusiones y Trabajo Futuro	9
6.1	Conclusiones	9
6.2	Trabajo Futuro	10
7	Anexos	10
7.1	Código fuente (snippets relevantes)	10
7.2	Tablas y gráficos con resultados de evaluación.	10
7.3	Glosario de términos técnicos.	10

7.4 Bibliografía y referencias.	10
---	----

1 Introducción

1.1 Contexto y Motivación

El curso de Introducción a la Programación de la carrera de Ingeniería Informática en la Universidad de Matanzas tiene como objetivo fundamental capacitar a los estudiantes en la solución de problemas mediante el uso de computadoras. El plan de estudios se estructura en torno a tres temas principales. Inicialmente, en el Tema I, se aborda la Algoritmización, donde los estudiantes aprenden a resolver problemas utilizando algoritmos informales. Posteriormente, en el Tema II, se introducen las Estructuras de control, cubriendo conceptos clave como tipos de datos, variables, constantes, asignación, expresiones, así como las estructuras de control alternativas y repetitivas, culminando con el estudio de funciones. Finalmente, en el Tema III, se exploran los Arreglos, tanto unidimensionales como bidimensionales, y se analizan algoritmos básicos para su manipulación. El curso proporciona una base sólida en los conceptos fundamentales de la programación, preparando a los estudiantes para abordar problemas computacionales de manera metódica y estructurada.

Los estudiantes que se inician en el aprendizaje de programación, y en particular al abordar el contenido específico del curso, a menudo se enfrentan a dificultades que pueden obstaculizar su comprensión y progreso. Uno de los principales desafíos surge de la dificultad para encontrar información específica dentro de los materiales del curso. Aunque las conferencias y guías de estudio proporcionan una base teórica, a menudo carecen de la granularidad necesaria para responder a preguntas puntuales que los estudiantes puedan tener al trabajar en ejercicios prácticos. Los conceptos clave pueden estar dispersos en varios documentos, lo que dificulta la localización de información específica cuando se necesita una referencia rápida o una aclaración sobre un detalle concreto. Esta dificultad en encontrar información puntual puede llevar a los estudiantes a dedicar un tiempo excesivo a la búsqueda, en detrimento del tiempo que dedican a la práctica.

Además, los estudiantes principiantes a menudo luchan con la necesidad de respuestas contextualizadas que estén alineadas con el contenido del curso. Aunque los materiales didácticos explican los conceptos fundamentales como las estructuras de control o los arreglos, los estudiantes necesitan ejemplos prácticos y aclaraciones que demuestren cómo aplicar estas herramientas a los problemas específicos planteados en los ejercicios y talleres del curso. Las respuestas generales o teóricas pueden resultar insuficientes para comprender cómo implementar un algoritmo de búsqueda en un arreglo unidimensional o cómo anidar estructuras de control para resolver un problema específico del curso. La falta de ejemplos y explicaciones que se vinculen directamente con el contenido y el nivel del curso puede dificultar que los estudiantes apliquen los conocimientos adquiridos de manera efectiva.

1.2 Objetivos del Proyecto

Objetivo general del proyecto: mejorar el acceso a la información y el soporte de aprendizaje en el curso de Introducción a la Programación de la Universidad de Matanzas mediante la implementación de un sistema de Generación Aumentada por Recuperación (RAG).

Objetivos específicos:

- Recopilar y organizar exhaustivamente todos los materiales del curso en un corpus digital.
- Diseñar e implementar un sistema RAG eficiente que integre un motor de búsqueda semántico y un modelo de lenguaje, asegurando respuestas relevantes y contextualizadas.
- Adaptar las respuestas generadas al contexto específico del curso, proporcionando ejemplos prácticos y explicaciones alineadas con los temas tratados.
- Facilitar la búsqueda de información puntual, permitiendo a los estudiantes encontrar rápidamente lo que necesitan.
- Evaluar y mejorar continuamente el sistema a través del feedback de usuarios y métricas de rendimiento.

1.3 Alcance del Proyecto

El alcance de este proyecto se centra en el desarrollo e implementación de un sistema de Generación Aumentada por Recuperación (RAG), específicamente diseñado para apoyar a los estudiantes del curso de Introducción a la Programación de la carrera de Ingeniería Informática de la Universidad de Matanzas. Este sistema actuará como una herramienta de aprendizaje que facilitará el acceso a información relevante y contextualizada, ayudando a los estudiantes a comprender y aplicar los conceptos clave del curso de manera más efectiva.

Materiales del Curso como Base de Conocimiento: El sistema RAG utilizará como base de conocimiento los materiales del curso, incluyendo las conferencias en formato PDF, las guías de estudio, los enunciados de ejercicios y talleres. Estos documentos, que abarcan la teoría y la práctica del curso, serán procesados para extraer su contenido textual, limpiarlo y tokenizarlo, generando así los embeddings que permitirán una búsqueda semántica eficiente. Adicionalmente, los materiales procesados se almacenarán en una base de datos vectorial, facilitando la recuperación rápida y precisa de la información relevante en respuesta a las preguntas de los estudiantes.

Tipos de Preguntas que el Sistema RAG Debe Responder: El sistema RAG estará diseñado para responder una amplia gama de preguntas, abarcando desde conceptos teóricos hasta la sintaxis de código y la resolución de problemas específicos del curso. Esto incluye consultas sobre definiciones y explicaciones de conceptos clave, preguntas sobre el funcionamiento de los algoritmos y las estructuras de datos, así como preguntas sobre la sintaxis y el uso de palabras reservadas de Java. Además, el sistema deberá ofrecer ejemplos de código que ilustren cómo aplicar los conceptos aprendidos, y ayudar a los estudiantes a abordar los problemas planteados en los ejercicios y talleres del curso, proporcionando explicaciones contextualizadas y ejemplos prácticos. Sin embargo, es importante destacar que el sistema debe estar diseñado para guiar el aprendizaje y ayudar al entendimiento, no para resolver los problemas por los estudiantes, ni para responder preguntas que estén fuera del ámbito del curso.

Usuarios Objetivo: Los usuarios principales de este sistema RAG son los estudiantes del curso de Introducción a la Programación. Se considerará que estos estudiantes son principiantes en el mundo de la programación, por lo que las respuestas proporcionadas

por el sistema serán claras, sencillas y evitarán el uso de jerga técnica innecesaria. Además, el sistema será accesible a través de una interfaz web intuitiva y fácil de usar, que permitirá a los estudiantes formular sus preguntas en lenguaje natural sin necesidad de tener conocimientos técnicos avanzados. La interacción con el sistema estará diseñada para ser lo más sencilla y natural posible, facilitando su uso por parte de todos los estudiantes del curso.

Énfasis en la Usabilidad: La usabilidad del sistema RAG será una prioridad clave. Se desarrollará una interfaz de usuario simple y fácil de usar, que permitirá a los estudiantes formular sus preguntas y recibir respuestas de manera eficiente. Las respuestas generadas por el sistema serán claras, concisas y fáciles de entender para los estudiantes principiantes. Además, el sistema incluirá un mecanismo para que los usuarios puedan proporcionar feedback sobre la calidad de las respuestas recibidas, lo que facilitará la mejora continua del sistema y su adaptación a las necesidades específicas de los estudiantes del curso.

2 Diseño del Sistema RAG

2.1 Arquitectura General del Sistema

El sistema RAG, cuya arquitectura se muestra en la **Figura 1**, se organiza como un flujo de procesamiento de información que integra el componente de Retrieval con el componente de Generation buscando responder a las consultas del Usuario de manera informada y contextualizada la arquitectura se compone de siete elementos principales representados en el diagrama el Usuario que inicia el proceso el Componente de Entrada (Input) que recibe la consulta el Componente de Recuperación (Retrieval) encargado de buscar información relevante el Almacén Vectorial (Vector Store) que gestiona los embeddings el Componente de Generación (Generation) que prepara el prompt el modelo de lenguaje Gemini y el Componente de Salida (Output) que presenta la respuesta El Usuario es el actor principal que inicia el proceso ingresando una consulta específica el Componente de Entrada (Input) recibe esta consulta preparándola para su procesamiento posterior este componente aplica pasos de limpieza normalización y tokenización El Componente de Recuperación (Retrieval) es responsable de buscar en la fuente de conocimiento la información relevante para la consulta procesada utilizando para ello el Almacén Vectorial (Vector Store) este componente realiza una búsqueda vectorial obteniendo los fragmentos más relevantes el Almacén Vectorial (Vector Store) es una base de datos especializada que gestiona las representaciones vectoriales de los Documentos almacenados en la fuente de conocimiento realizando la indexación búsqueda y almacenamiento de los embeddings el Componente de Generación (Generation) actúa como puente entre el Componente de Recuperación (Retrieval) y el modelo de lenguaje Gemini combinando la consulta original con los fragmentos recuperados para construir un prompt para el modelo Gemini el modelo Gemini utiliza su conocimiento interno y el contexto del prompt para generar una respuesta coherente y relevante el Componente de Salida (Output) recibe esta respuesta formateándola y presentándola al Usuario el flujo de datos comienza con la consulta del Usuario que pasa por el Componente de Entrada (Input) luego al Componente de Recuperación (Retrieval) que consulta al Almacén Vectorial (Vector Store) este retorna fragmentos que son procesados por el Componente de Recuperación (Retrieval) y enviados al Componente de Generación (Generation) que genera el prompt para el modelo Gemini

que produce la respuesta que el Componente de Salida (Output) presenta al Usuario de esta forma el sistema RAG aprovecha las capacidades de los LLMs como el modelo Gemini para generar texto y la búsqueda semántica del Almacén Vectorial ofreciendo resultados más precisos y contextualizados la estructura modular permite que cada componente sea adaptado a las necesidades del dominio y a los objetivos específicos del sistema.

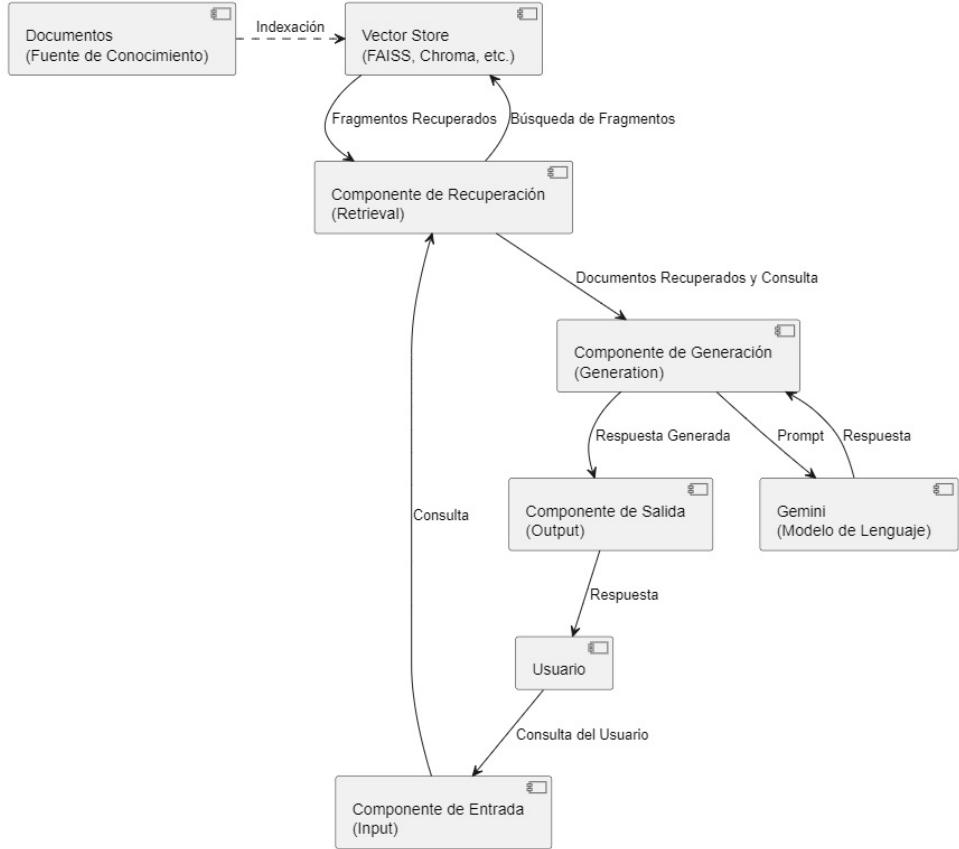


Figure 1: Diagrama de la arquitectura del sistema RAG (**Figura 1**)

Este diagrama de secuencia **Figura 2** ilustra el funcionamiento de un chatbot de inteligencia artificial basado en la técnica RAG (Retrieval-Augmented Generation) que opera a través de Telegram. El usuario inicia la interacción enviando un mensaje al bot, que a su vez lo reenvía a una API Gateway. Esta API Gateway dirige la solicitud a un Back-end Service, que se encarga de la lógica RAG. El Back-end Service primero consulta un Vector Store para obtener fragmentos relevantes de información según la pregunta del usuario. Luego, utiliza esta información contextual, junto con la pregunta original, para enviarla a un Language Model, el cual genera una respuesta. La respuesta es devuelta al Back-end Service, que a su vez la envía a la API Gateway, y esta última la transmite al bot de Telegram. Finalmente, el bot de Telegram entrega la respuesta generada al usuario. Este diagrama destaca la separación de responsabilidades entre los componentes: el bot como interfaz, el API Gateway como punto de entrada, el Back-end Service con la lógica RAG, el Vector Store para la recuperación de información, y el Language Model para la generación de respuestas. En resumen, el diagrama muestra el flujo completo de un sistema RAG, desde la interacción del usuario hasta la generación de una respuesta informada y contextual.

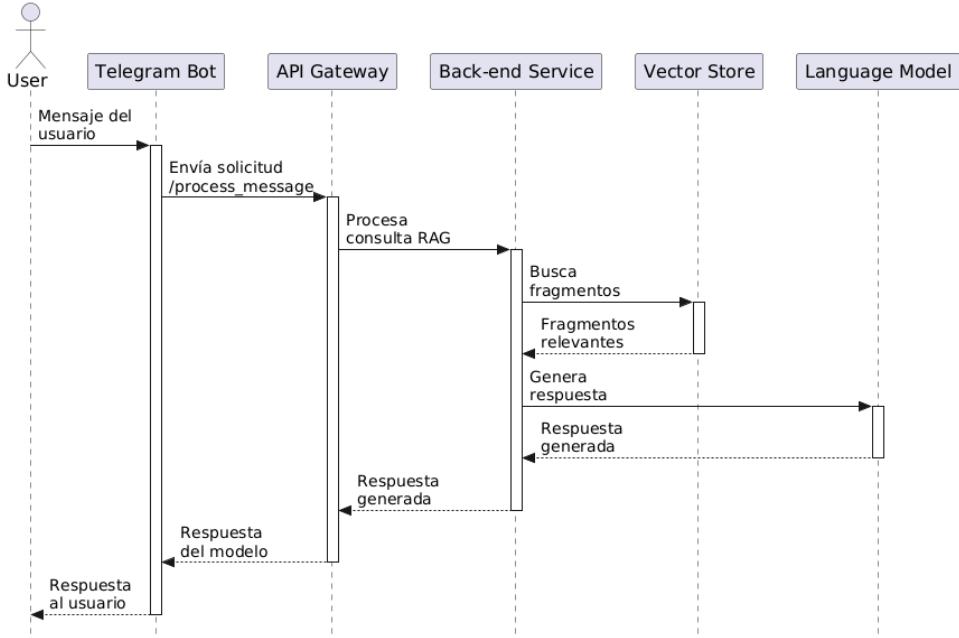


Figure 2: Diagrama de secuencia del bot (**Figura 2**)

2.2 Preprocesamiento de los Documentos

- Extracción de texto de los PDFs (bibliotecas utilizadas: PyPDF2, pdfplumber, Tesseract OCR).
- Limpieza y normalización del texto (eliminación de caracteres especiales, minúsculas, lematización, etc.).
- Fragmentación del texto en unidades semánticas (párrafos, secciones, etc.).

2.3 Modelo de Recuperación

- Selección del modelo de embeddings (Sentence Transformers, OpenAI Embeddings, otros).
- Justificación de la elección del modelo.
- Generación de embeddings para los fragmentos de texto.
- Creación del índice de búsqueda vectorial (Faiss, Annoy, bases de datos vectoriales).
- Optimización del índice.

2.4 Modelo de Generación

- Selección del LLM (OpenAI GPT-3.5/4, Llama, Mistral, otros).
- Justificación de la elección del LLM.
- Diseño del prompt para el LLM para la utilización de la información recuperada.

2.5 Integración del Modelo de Recuperación y Generación

- Descripción del proceso de recuperación de fragmentos relevantes.
- Descripción del proceso de aumento de la consulta con la información recuperada.
- Descripción del proceso de generación de la respuesta por parte del LLM.

3 Implementación del Sistema

3.1 Herramientas y Tecnologías Utilizadas

- Bibliotecas y frameworks de Python utilizados: (Transformers, Faiss/Annoy, LangChain/LlamaIndex).
- Bases de datos vectoriales utilizadas: (si aplica)
- Entorno de desarrollo: (IDE, sistema operativo).

3.2 Desarrollo del Pipeline de RAG

- Pasos detallados para la implementación del pipeline (carga de documentos, preprocesamiento, generación de embeddings, indexación, búsqueda, generación de respuesta).
- Fragmentos de código relevantes (ejemplos).

Listing 1: Ejemplo de preprocesamiento

```
def preprocess_text(text):
    text = text.lower()
    # ... otros pasos ...
    return text
```

3.3 Despliegue del Sistema (si aplica)

- Descripción del entorno de despliegue (servidor, nube, etc.)
- Tecnologías y herramientas utilizadas para el despliegue (Docker, etc.).

4 Evaluación del Sistema

4.1 Diseño de la Evaluación

- Definición de las métricas de evaluación:
 - Relevancia de los documentos recuperados.
 - Precisión y calidad de las respuestas generadas.
 - Fluidez y coherencia de las respuestas.
 - Cobertura de las respuestas.

- Metodología de evaluación:
 - Creación de un conjunto de datos de Ground Truth.
 - Pruebas con usuarios reales.

4.2 Resultados de la Evaluación

- Presentación de los resultados de la evaluación usando las métricas definidas.
 - Presentación de los resultados de la evaluación usando las métricas definidas.
 - Análisis de los puntos fuertes y débiles del sistema.
 - Identificación de áreas de mejora.

5 Análisis y Discusión

5.1 Beneficios del Sistema RAG

- Mejora del acceso a la información y la comprensión de conceptos.
- Reducción del tiempo de búsqueda y solución de dudas.
- Aumento de la autonomía del estudiante.
- Potencial para personalizar el aprendizaje.

5.2 Desafíos y Limitaciones

- Dificultades técnicas encontradas durante el desarrollo.
- Limitaciones del sistema RAG (comprensión de preguntas ambiguas, posibles errores en las respuestas).
- Potenciales sesgos y limitaciones en el conjunto de datos y los modelos.
- Recomendaciones para superar estos desafíos.

6 Conclusiones y Trabajo Futuro

6.1 Conclusiones

- Resumen de los logros del proyecto.
- Respuesta a los objetivos planteados.
- Reflexión sobre el impacto del sistema en el proceso de aprendizaje.

6.2 Trabajo Futuro

- Propuestas para mejorar el sistema RAG (fine-tuning, nuevos modelos, mejorar la calidad de datos).
- Posibles extensiones del sistema (integración con otras herramientas, personalización más avanzada).
- Investigaciones futuras.

7 Anexos

7.1 Código fuente (snippets relevantes)

7.2 Tablas y gráficos con resultados de evaluación.

7.3 Glosario de términos técnicos.

7.4 Bibliografía y referencias.