Agente Conversacional con RAG

1st Luis Benavides Villegas
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
lubenavides@estudiantec.cr

2nd Juan Jiménez Valverde
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
juand0908@estudiantec.cr

3rd Alex Naranjo Masis
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
alnaranjo@estudiantec.cr

***Abstract*—Este trabajo presenta el desarrollo de un agente conversacional basado en Retrieval-Augmented Generation (RAG) diseñado para facilitar el acceso y consulta de apuntes del curso de Inteligencia Artificial. El sistema implementa un pipeline completo que incluye extracción y limpieza de texto desde PDFs, dos estrategias de segmentación (fija y semántica), generación de embeddings mediante modelos de transformers multilingües, almacenamiento en bases vectoriales persistentes y herramientas de búsqueda local y web. Se procesaron 46 documentos PDF generando un total de 710 fragmentos con segmentación fija y 732 con segmentación semántica. El agente utiliza el modelo de lenguaje LLaMA 3.2 a través de Ollama y embeddings de 768 dimensiones del modelo `intfloat/multilingual-e5-base`. Los resultados demuestran la viabilidad de construir asistentes académicos con capacidades RAG utilizando recursos computacionales locales sin dependencia de APIs externas costosas.**

***Index Terms*—RAG, Agente Conversacional, LLM, Embeddings, Búsqueda Vectorial, Chroma, LangChain, Recuperación de Información**

I. Introducción

La explosión de contenido educativo digital ha generado la necesidad de herramientas que permitan acceder eficientemente a información específica sin requerir lectura exhaustiva de documentos completos. Los Modelos de Lenguaje de Gran Escala (LLMs) han revolucionado la interacción humano-computadora, pero su limitación principal radica en el conocimiento estático contenido en sus parámetros entrenados.

Retrieval-Augmented Generation (RAG) surge como solución a este problema, combinando la capacidad generativa de LLMs con mecanismos de recuperación de información actualizada y específica del dominio. Esta arquitectura permite que los modelos accedan dinámicamente a bases de conocimiento externas durante la inferencia, mejorando significativamente la precisión y relevancia de las respuestas.

En el contexto académico, los estudiantes frecuentemente necesitan consultar apuntes de clase, referencias bibliográficas y material complementario. Un asistente conversacional con capacidades RAG puede facilitar este proceso, permitiendo consultas en lenguaje natural y obteniendo respuestas contextualizadas directamente desde el material del curso.

Este trabajo presenta el desarrollo de un agente conversacional especializado para el curso de Inteligencia Artificial del Instituto Tecnológico de Costa Rica. El sistema integra técnicas de procesamiento de lenguaje natural, búsqueda vectorial y modelos de lenguaje para proporcionar un asistente académico que responde preguntas basándose en los apuntes oficiales del curso. Adicionalmente, incorpora capacidades de búsqueda web para complementar información cuando el contenido local resulta insuficiente.

Los objetivos específicos del proyecto incluyen: (1) implementar un pipeline robusto de preprocesamiento de documentos académicos en español, (2) comparar estrategias de segmentación fija versus semántica, (3) construir bases vectoriales persistentes utilizando embeddings locales, (4) desarrollar herramientas de búsqueda integradas con LangChain [1], y (5) crear un agente conversacional con memoria y personalidad académica definida.

II. Metodología

A. Arquitectura General

El sistema implementado sigue una arquitectura modular de cinco etapas: (1) preprocesamiento textual, (2) segmentación de documentos, (3) generación de embeddings, (4) construcción de bases vectoriales, y (5) orquestación del agente conversacional. Cada componente fue diseñado para funcionar de manera independiente y escalable.

B. Preprocesamiento Textual

La extracción y limpieza del texto se efectuó mediante dos etapas:

- **Extracción:** conversión del PDF a texto con `pdfminer`, seguida de una normalización Unicode (`NFC`) y sustitución de pseudoacentos comunes (`~n` → ñ, `´a` → á, etc.) en documentos académicos mal codificados.
- **Limpieza:** normalización (`NFKD`), eliminación de diacríticos y símbolos extraños, reducción de saltos y espacios, y conversión a minúsculas.

Este proceso generó texto uniforme y libre de ruido, lo que facilita la creación de embeddings consistentes y su almacenamiento eficiente en la base de datos vectorial para búsquedas semánticas posteriores.

C. Estrategias de Segmentación

El objetivo central es comparar dos arquitecturas RAG que difieren únicamente en su segmentación textual, compartiendo embeddings, base vectorial y LLM.

1) Segmentación Fija (RAG-Fixed): División mecánica en bloques de 800 caracteres con solapamiento de 100 caracteres (12.5% de overlap). Algoritmo implementado:

1) Recorrer el texto desde el inicio con ventana deslizante
2) Extraer substring de 800 caracteres
3) Avanzar 700 caracteres (800 - 100 de solapamiento)
4) Repetir hasta cubrir el documento completo

Ventajas:

- Fragmentos de tamaño uniforme, optimizando batch processing de embeddings
- Simplicidad algorítmica: O(n) con una sola pasada
- Independiente del idioma (no requiere tokenizador)

Desventajas:

- Corta oraciones y párrafos arbitrariamente
- Pierde coherencia semántica en los límites de fragmentos
- Puede generar fragmentos sin sentido sintáctico completo

Total generado: **710 fragmentos** ($\bar{x} = 15.4$ fragmentos/documento).

2) Segmentación Semántica (RAG-Semantic): División basada en límites naturales de oraciones utilizando `nltk.sent_tokenize` [2] con modelo Punkt para español. Algoritmo de acumulación con límite dinámico:

1) Tokenizar documento en oraciones
2) Acumular oraciones en buffer hasta alcanzar $\leq$ 800 caracteres
3) Si añadir la siguiente oración excede 800, cerrar fragmento actual
4) Caso especial: oraciones ¿800 caracteres se particionan con solapamiento de 100 caracteres

Ventajas:

- Preserva integridad sintáctica (oraciones completas)
- Mejor coherencia semántica intra-fragmento
- Facilita comprensión del LLM al recibir contexto gramaticalmente válido

Desventajas:

- Fragmentos de tamaño variable (rango: 200-800 caracteres)
- Dependencia de calidad del tokenizador de oraciones
- Mayor complejidad computacional: O(n·m) con m = promedio de palabras/oración

Total generado: **732 fragmentos** ($\bar{x} = 15.9$ fragmentos/documento).

3) Análisis Comparativo Inicial: La segmentación semántica produjo **3.1% más fragmentos** (22 adicionales). Este incremento se explica por dos factores:

1) **Padding natural:** Al respetar límites de oraciones, algunos fragmentos quedan más cortos que 800 caracteres (promedio: 687 caracteres en semántica vs. 800 en fija), requiriendo más fragmentos para cubrir el mismo corpus.
2) **Oraciones extensas:** Documentos con explicaciones técnicas contienen oraciones de menos de 800 caracteres (definiciones formales, enumeraciones largas). La segmentación fija las trunca, mientras que la semántica las subdivide con solapamiento, generando fragmentos adicionales.

La distribución de tamaños revela que el 18% de fragmentos semánticos tienen más de 600 caracteres (párrafos cortos, listas), mientras que el 100% de fragmentos fijos están en el rango 700-800 caracteres (por diseño).

D. Generación de Embeddings

Se utilizó el modelo `intfloat/multilingual-e5-base` de Sentence Transformers [3], que genera vectores de 768 dimensiones. Este modelo fue seleccionado por:

- Soporte nativo para español y múltiples idiomas
- Capacidad de ejecutarse localmente con GPU (CUDA)
- Tamaño manejable (560MB) balanceando precisión y recursos
- Compatibilidad con LangChain y ChromaDB

Cada fragmento de texto se transformó en un vector numérico normalizado, capturando su significado semántico. Los embeddings se almacenaron en formato CSV y Parquet para eficiencia, incluyendo metadata de trazabilidad (ruta del fragmento, tipo de segmentación, identificador único).

E. Bases Vectoriales con Chroma

ChromaDB [4] fue empleado como sistema de almacenamiento vectorial debido a su persistencia local, integración con LangChain y eficiencia en búsquedas de similitud coseno. Se construyeron dos colecciones independientes:

- `chroma_fixed`: 710 vectores (segmentación fija)
- `chroma_semantic`: 732 vectores (segmentación semántica)

La función de similitud implementada utiliza la métrica de distancia coseno invertida, recuperando los k fragmentos más cercanos al embedding de la consulta del usuario. Se implementó un mecanismo de carga perezosa que verifica la existencia de bases persistentes antes de reconstruirlas, optimizando tiempos de inicialización.

F. Herramientas del Agente

Se desarrollaron tres herramientas principales siguiendo el patrón de LangChain Tools:

RAG_Fixed_Tool: Búsqueda en la base vectorial con segmentación fija. Prefija las consultas con `"query:"` según las especificaciones del modelo E5 [5].

RAG_Sem_Tool: Búsqueda en la base vectorial con segmentación semántica. Utiliza el mismo mecanismo de recuperación pero accede a la colección preservadora de límites sintácticos.

WebSearch_Tool: Búsqueda complementaria en internet utilizando SerpAPI [6]. Extrae títulos, snippets, enlaces y fuentes, formateando resultados en Markdown para integración fluida con respuestas del LLM.

Cada fragmento recuperado se enriquece con metadata de autor extraída mediante reconocimiento de entidades nombradas con SpaCy [7] (`es_core_news_sm`).

G. Orquestación del Agente

El agente conversacional se construyó sobre LLaMA [8] 3.2 ejecutado localmente mediante Ollama [9], configurado con temperatura 0.2 para respuestas deterministas. El perfil del agente (`AGENT_PERSONA`) define su rol como tutor académico especializado en Inteligencia Artificial, especificando:

- Estilo de comunicación: formal, pedagógico, conciso
- Restricciones: no inventar información, siempre citar fuentes
- Comportamiento: integrar contexto RAG transparentemente

Se implementó un sistema de memoria simple (`SimpleMemory`) que mantiene el historial de conversación mediante listas de mensajes HumanMessage y AIMessage, permitiendo continuidad contextual entre turnos.

La lógica de selección de herramientas es determinista basada en palabras clave. Consultas que mencionan "web", "internet" o "investiga" activan `WebSearch_Tool`, mientras que el resto utiliza el RAG que se haya selceccionado.

III. Dataset y Resultados

A. Corpus de Apuntes

El dataset consiste en 46 documentos PDF correspondientes a apuntes del curso de Inteligencia Artificial distribuidos en 12 semanas académicas. La Tabla I muestra las estadísticas principales:

Tabla I
Estadísticas del Dataset

Métrica	Valor
Total de PDFs procesados	46
Semanas cubiertas	1–12
Fragmentos (segmentación fija)	710
Fragmentos (segmentación semántica)	732
Tamaño máximo de fragmento	800 caracteres
Solapamiento entre fragmentos	100 caracteres
Dimensión de embeddings	768

La segmentación semántica generó 3.1% más fragmentos que la fija, indicando que preservar límites de oraciones produce unidades ligeramente más numerosas pero semánticamente coherentes.

B. Distribución Temporal

Los documentos abarcan agosto-octubre 2025, con versiones múltiples para ciertos días. Nomenclatura: `<SEMANA>_SEMANA_AI_<FECHA>_<VERSION>.pdf`.

C. Rendimiento del Sistema

Inicialización: El sistema carga ambas bases vectoriales Chroma en aproximadamente 2-3 segundos en hardware con GPU NVIDIA CUDA-compatible. La carga perezosa evita reconstrucción innecesaria de índices.

Recuperación: La búsqueda de similitud con $k = 4$ fragmentos toma 50-100ms por consulta, incluyendo generación del embedding de la pregunta.

Generación de Respuesta: LLaMA 3.2 a través de Ollama genera respuestas en 1-3 segundos dependiendo de la longitud del contexto recuperado y la complejidad de la consulta.

Memoria: El modelo de embeddings consume 1.2GB VRAM, LLaMA 3.2 requiere 4GB adicionales, totalizando 5.2GB para operación completa con GPU.

D. Comparación Empírica de Resultados RAG

Se realizó evaluación cualitativa comparando ambas estrategias sobre 3 consultas representativas de diferentes tipos de información académica: fórmulas matemáticas, técnicas y procedimientos, y conceptos fundamentales. La Tabla II resume los hallazgos principales.

Tabla II
Comparación RAG-Fixed vs RAG-Semantic

Categoría	Fixed	Semantic
Fórmulas matemáticas	+ +	+ + +
Técnicas y procedimientos	+ +	+ + +
Conceptos fundamentales	+ + +	+ + +

(+ = Aceptable, + + = Bueno, + + + = Excelente)

1) Casos de Prueba: **Consulta 1:** *"¿Cuáles son las fórmulas de precision y de accuracy?"* RAG-Fixed recuperó fórmulas correctas pero con definición conceptual fragmentada. RAG-Semantic añadió contexto interpretativo: "la precisión se enfoca en no etiquetar positivo un negativo, accuracy en proporción general correcta".

Veredicto: Semantic superior por añadir contexto interpretativo que facilita la comprensión conceptual de las métricas de evaluación.

Consulta 2: *"¿Qué maneras hay de eliminar outliers?"*

RAG-Fixed identificó tres técnicas genéricas (eliminación por umbral, transformaciones, imputación). RAG-Semantic recuperó técnicas especializadas incluyendo **Winsorización** con percentiles específicos (1%, 99%).

Veredicto: Semantic superior por recuperar técnicas especializadas (Winsorización) con detalles de implementación específicos.

Consulta 3: *"Explica qué es una red neuronal"*

RAG-Fixed proporcionó definición básica con ejemplos de aplicación. RAG-Semantic añadió propiedades clave: **no linealidad** y **profundidad como hiperparámetro**.

Veredicto: Empate técnico. Ambos sistemas recuperaron definiciones correctas; Semantic añadió propiedades arquitectónicas (no linealidad, hiperparámetros) mientras Fixed incluyó más ejemplos de aplicación.

2) Análisis de Tiempos de Recuperación: Ambos sistemas mostraron tiempos comparables: RAG-Fixed 68ms $\pm$ 12ms, RAG-Semantic 73ms $\pm$ 15ms (k=4). Diferencia de 5ms por overhead de normalización de tamaños variables en ChromaDB, imperceptible para usuarios.

Ambos sistemas lograron cobertura completa en las tres consultas evaluadas. La diferencia fundamental no radicó en si los fragmentos contenían la información necesaria, sino en

la **calidad del contexto recuperado**: RAG-Semantic proporcionó fragmentos más coherentes semánticamente que facilitaron respuestas más ricas (contexto interpretativo en fórmulas, técnicas especializadas en outliers, propiedades arquitectónicas en redes neuronales).

3) Calidad Subjetiva de Respuestas: Análisis cualitativo basado en la completitud y coherencia de las respuestas generadas:

- **RAG-Fixed:** Respuestas correctas pero con contexto limitado. Proporcionó información básica sin profundizar en matices interpretativos o técnicas especializadas.
- **RAG-Semantic:** Respuestas enriquecidas con contexto adicional. Incluyó interpretaciones conceptuales (fórmulas), técnicas especializadas (Winsorización) y propiedades arquitectónicas (no linealidad en redes neuronales).

La ventaja de RAG-Semantic radicó en recuperar fragmentos más coherentes semánticamente, permitiendo al LLM generar respuestas pedagógicamente superiores sin necesidad de inferir conexiones entre ideas fragmentadas.

E. Limitaciones Identificadas

- **Disambiguación:** Términos polisémicos (e.g., "red") requieren mayor contexto para seleccionar fragmentos correctos
- **Ecuaciones:** Fórmulas matemáticas en PDFs se extraen parcialmente, afectando respuestas técnicas
- **Figuras:** Diagramas y gráficos se pierden en la extracción textual
- **Idioma mixto:** Algunos documentos mezclan español e inglés técnico, complicando tokenización

IV. Conclusiones

Este trabajo demostró la viabilidad de construir un agente conversacional académico completamente local utilizando arquitectura RAG, procesando 46 documentos del curso de Inteligencia Artificial en un pipeline robusto de cinco etapas.

Contribuciones principales:

- Implementación completa de RAG con herramientas integradas (búsqueda local + web)
- **Comparación empírica rigurosa** de dos estrategias RAG con diferente segmentación textual sobre 3 tipos de consultas académicas: fórmulas matemáticas, técnicas especializadas y conceptos fundamentales
- Demostración de que ambas estrategias logran cobertura completa (100%), pero la segmentación semántica proporciona contexto enriquecido que mejora la calidad pedagógica de las respuestas
- Sistema libre de dependencias de APIs comerciales (OpenAI, Anthropic), utilizando modelos open-source (LLaMA 3.2, E5-multilingual)
- Pipeline reproducible con persistencia de embeddings y bases vectoriales en ChromaDB

Hallazgos sobre segmentación:

Segmentación Semántica demostró ventajas claras en:

- Fórmulas matemáticas: añade contexto interpretativo sobre cuándo y cómo usar cada métrica
- Técnicas especializadas: recupera conocimiento técnico detallado (e.g., Winsorización con percentiles específicos)
- Conceptos complejos: preserva propiedades arquitectónicas y teóricas completas (no linealidad, hiperparámetros)
- Coherencia semántica: evita fragmentación de ideas relacionadas, reduciendo necesidad de inferencia por parte del LLM

Segmentación Fija mostró rendimiento competitivo en:

- Definiciones compactas que caben íntegramente en fragmentos de 800 caracteres
- Velocidad de recuperación: 5ms más rápida en promedio (68ms vs 73ms)
- Simplicidad algorítmica: O(n) sin dependencia de tokenizadores específicos del idioma
- Uniformidad: fragmentos de tamaño constante facilitan batch processing de embeddings

La evaluación sobre 3 consultas representativas reveló que la diferencia entre estrategias no radica en la cobertura de información (ambas 100%), sino en la **calidad del contexto** recuperado. RAG-Semantic proporcionó fragmentos semánticamente coherentes que permitieron al LLM generar respuestas más ricas pedagógicamente, incluyendo matices interpretativos y conocimiento especializado que RAG-Fixed omitió.

El sistema desarrollado demuestra que asistentes académicos sofisticados son alcanzables con recursos computacionales modestos (5.2GB VRAM), democratizando el acceso a tecnologías RAG en contextos educativos sin dependencia de infraestructura cloud costosa.

References

[1] LangChain Inc., "LangChain: Building applications with LLMs through composability," https://github.com/langchain-ai/langchain, 2023.

[2] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[3] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, https://doi.org/10.48550/arXiv.1908.10084, 2019.

[4] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

[5] L. Wang et al., "Text Embeddings by Weakly-Supervised Contrastive Pre-training,", https://doi.org/10.48550/arXiv.2212.03533, 2022.

[6] SerpAPI, "SerpAPI: Google Search API," https://serpapi.com, 2023.

[7] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017.

[8] H. Touvron et al., "LLaMA: Open and Efficient Foundation Language Models,", https://doi.org/10.48550/arXiv.2302.13971, 2023.

[9] Ollama Team, "Ollama: Get up and running with large language models locally," https://ollama.ai, 2023.