

# INVESTIGATHON - DOTANDO DE MEMORIA A LLMS APARTIR DE RAGS Y CHUNKING

Aleman Diego, Brostein Sacha, Hoyo Correa Bruno

Facultad de Ciencias Exactas y Naturales,Universidad de Buenos Aires



## INTRODUCCIÓN

Los agentes basados en modelos de lenguaje están migrando hacia arquitecturas más pequeñas, rápidas y portables. Para que estos sistemas puedan funcionar como asistentes útiles y persistentes, necesitan módulos de memoria capaces de recuperar información pasada de manera precisa, eficiente y con bajo costo computacional.

En los sistemas basados en LLMs, los fallos no provienen del modelo de lenguaje sino del sistema de recuperación: consultas poco informativas, embeddings ruidosos y mecanismos básicos de ranking limitan la capacidad de acceder al contexto relevante. Este trabajo se centra en mejorar el rendimiento de modelos pequeños (por debajo de los cuatro mil millones parámetros) mediante ajustes en la etapa de **indexing**, en particular explorando diferentes estrategias de **chunking**. Mostramos que pequeñas modificaciones en cómo se segmenta y organiza la memoria pueden producir mejoras simples, reproducibles y de bajo costo en precisión, velocidad y estabilidad del sistema.

## ¿QUÉ ES EL CHUNKING?

En este proyecto, **chunking** se refiere al proceso de dividir mensajes largos en fragmentos de longitud fija (medidos en caracteres) antes de indexarlos. Cada fragmento (*chunk*) se convierte en un embedding y se almacena en un índice que luego será consultado por el módulo de recuperación.

El tamaño del chunk determina:

- cuánta información contiene cada fragmento,
- cuántos chunks se generan por conversación,
- cuántos fragmentos deben recuperarse y enviarse al modelo,
- y cuán específico o ruidoso es cada embedding.

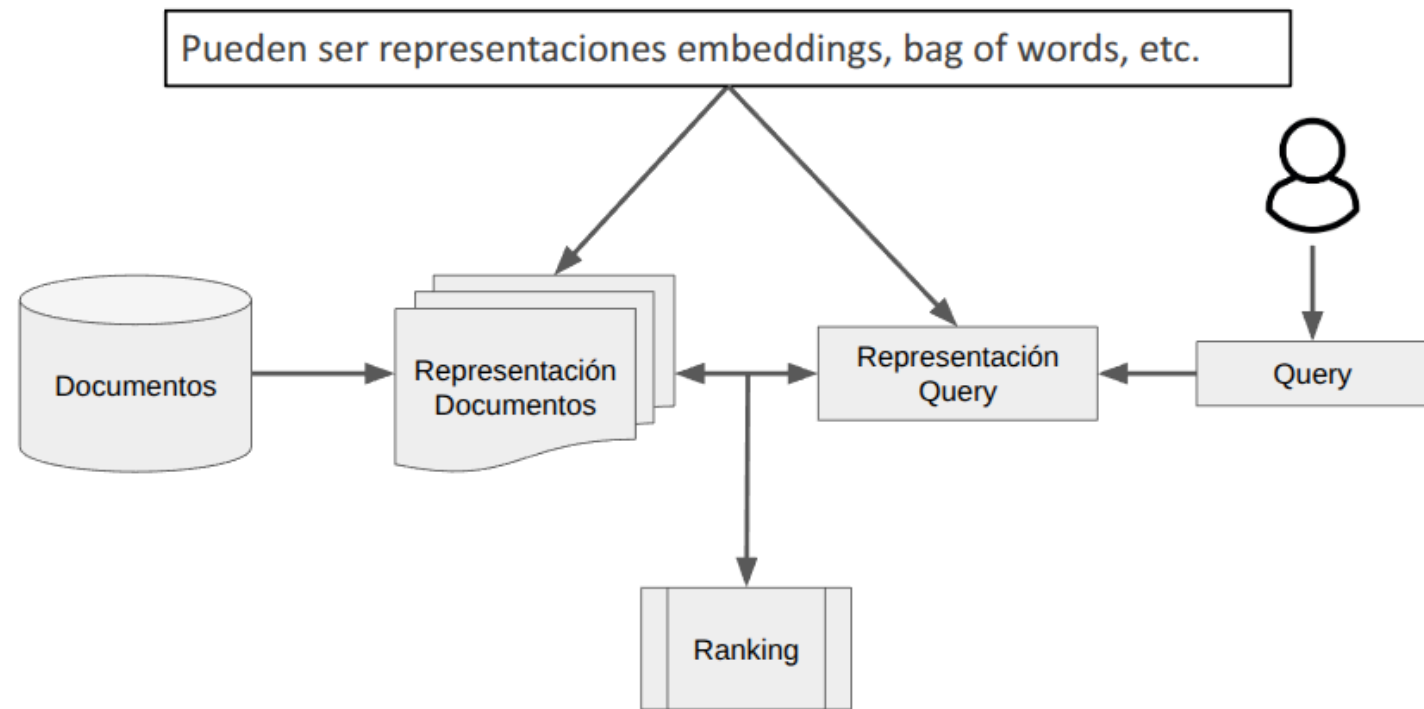
Estudiamos tamaños entre 250 y 5000 caracteres, analizando su impacto en precisión, latencia y longitud del contexto. .

## PIPELINE RAG

El pipeline evalúa un agente RAG sobre múltiples configuraciones de chunking siguiendo el marco de *LongMemEval*. El sistema completo incluye:

- **Indexing**: segmentación del historial, generación de embeddings y almacenamiento en un índice.
- **Retrieval**: recuperación de los chunks más relevantes mediante búsqueda densa.
- **Reading**: el modelo pequeño recibe el contexto recuperado y genera la respuesta final.

Nuestro análisis se enfoca exclusivamente en la etapa de **indexing**, permitiendo aislar el efecto directo del chunking en la precisión, el costo de contexto y la estabilidad del sistema.



## METODO DE EVALUACIÓN

Para estudiar el impacto del tamaño de los chunks en un pipeline RAG, implementamos un *RAG Agent* sobre el benchmark *LongMemEval* y desarrollamos un módulo propio de análisis. Para evaluar cada configuración de chunking ejecutamos el agente sobre todo el conjunto de preguntas y registramos las respuestas generadas y analizamos cuatro métricas clave en cada experimento:

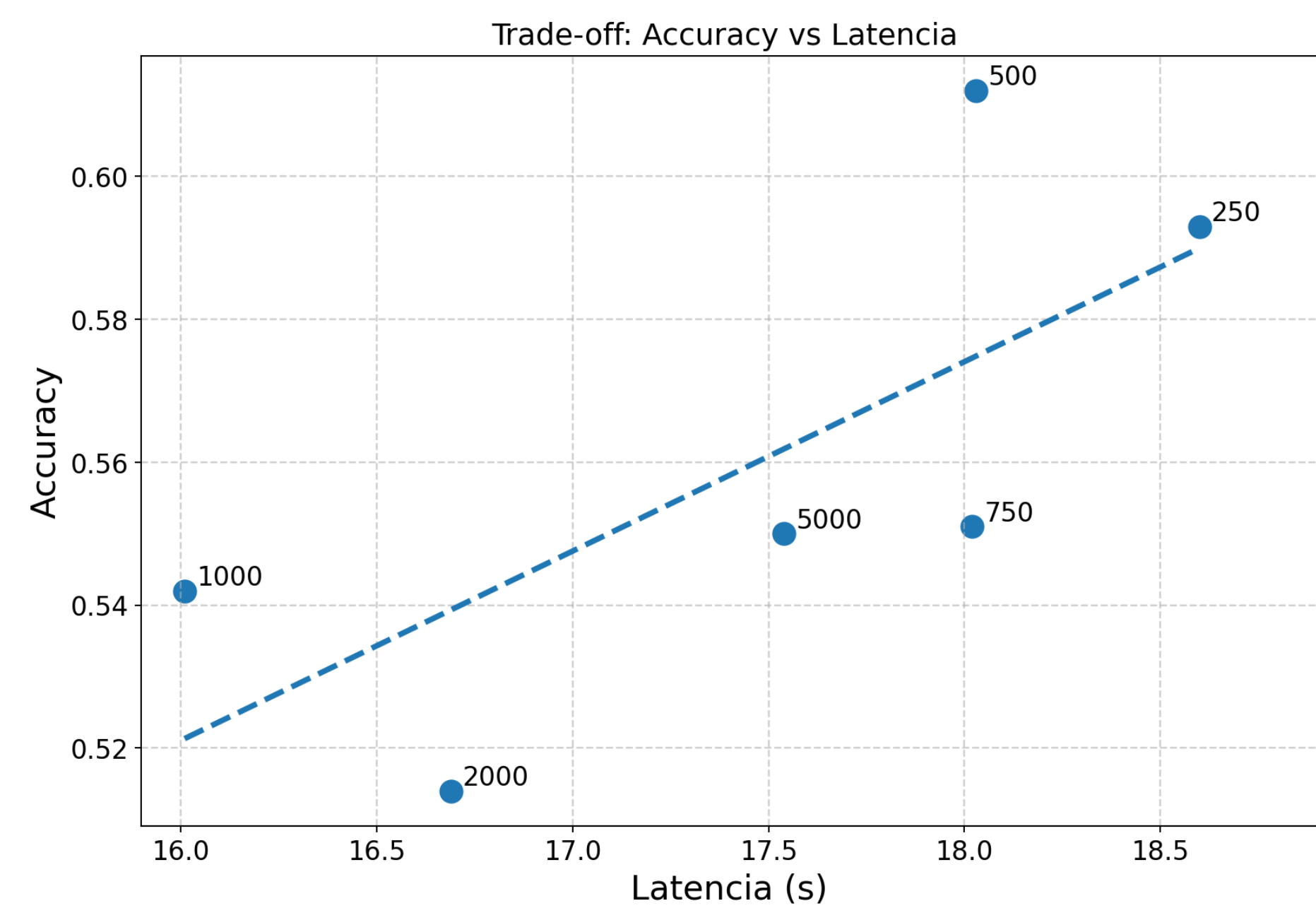
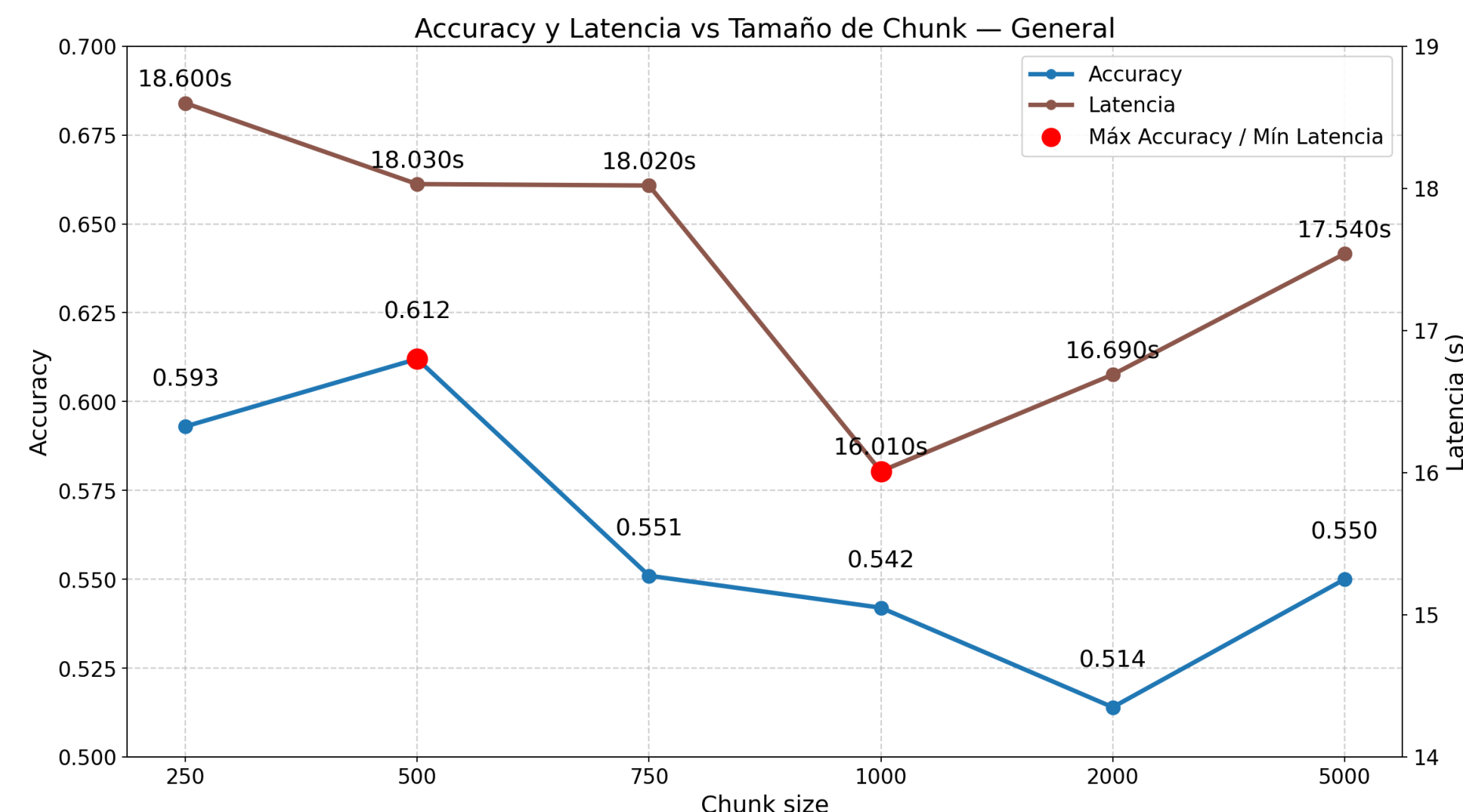
- **Exactitud**: coincidencia exacta entre la respuesta generada y la respuesta objetivo.
- **Latencia**: tiempo total requerido para responder cada query.
- **Longitud del contexto**: cantidad de caracteres/tokens enviados al LLM.
- **Retrieval**: número de chunks recuperados y posición del chunk relevante en el ranking.

Agrupamos los resultados según el tipo de pregunta: factuales cortas, preguntas que requieren combinar evidencias, y preguntas que dependen de información distante en la conversación.

## ANÁLISIS CUANTITATIVO Y EXPLORACIÓN VISUAL

El análisis conjunto reveló que el **chunking** es la principal palanca para resolver el *trade-off* entre la estabilidad y la calidad del sistema, con la Latencia inversamente proporcional al tamaño.

- **Costo Operacional**: La Latencia Media cae drásticamente desde **16.0 s** (250 car.) a **6.0 s** (5000 car.). Esto demuestra que los chunks pequeños multiplican las llamadas al índice, haciendo el sistema inestable y excesivamente lento.
- **Fallo del Extremo Rápido**: Los chunks más grandes (**5000 car.**) logran la Latencia mínima (6.0 s), pero sacrifican la Exactitud (40%), probando que la optimización ciega por velocidad destruye la calidad.



## CONCLUSIÓN

Nuestros resultados establecen que el **tamaño del chunk** no es solo un factor de precisión, sino una variable crítica para la **estabilidad operativa** y la **estrategia de rendimiento** de un agente RAG.

- **Trade-off Crítico**: La relación entre precisión y latencia es **no monótona** y define el *trade-off* funcional:
  - El punto de **máxima precisión** (0.612) se obtiene con 500 car., pero a costa de la latencia más alta (18.03 s). Prioriza la calidad.
  - El punto de **máxima velocidad** (16.01 s) se obtiene con 1000 car., con una precisión moderada (0.542). Prioriza la eficiencia.
- **El Riesgo de la Inestabilidad**: La opción de 2000 car. es **dominada**, ya que ofrece el peor *accuracy* (0.514) sin mejorar la latencia (16.69 s), confirmando que las configuraciones no óptimas provocan inestabilidad funcional.
- **Hallazgo Clave**: La decisión de *chunking* define si el agente actúa como un asistente preciso pero lento (500 car.) o como un sistema rápido y funcional (1000 car.).

Esto confirma que la **memoria efectiva no depende únicamente del modelo**, sino de cómo estructuramos el contexto en la etapa de *indexing*, permitiendo al ingeniero definir la estrategia de rendimiento del agente.

## Bibliografía

- Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., & Yu, D. (2024). *LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory*.
- Zhong, W., Guo, L., Gao, Qiqi., Ye, H., & Wang, Y. (2024). *MemoryBank: Enhancing Large Language Models with Long-Term Memory*.