Escribe aquí
tu frase favorita.

E indica aquí su autor

Agradecimientos

Me gustaría agradecer...

También quiero destacar...

Por último...

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

Introducción

1

Estrategias de chunkerización para LLMs

2

2.1. Introducción

La chunkerización en el ámbito de los modelos de lenguaje de gran escala (LLM) se refiere al proceso de dividir textos extensos en segmentos más manejables, conocidos como *chunks*. Esta técnica es fundamental para mejorar la gestión y el procesamiento de grandes volúmenes de información en aplicaciones de inteligencia artificial que utilizan modelos de lenguaje para entender y generar texto humano.

Necesidad de la Chunkerización: Los modelos de lenguaje, son inherentemente limitados por la cantidad de texto que pueden procesar en una sola instancia debido a restricciones de memoria y capacidad de cálculo. Al descomponer los textos en chunks más pequeños, se facilita que el modelo maneje datos extensos de manera eficiente, permitiendo una evaluación más rápida y precisa.

Objetivos de la Chunkerización: El principal objetivo de la chunkerización es maximizar la relevancia y la precisión del texto procesado. Al dividir el contenido en partes significativas y manejables, se busca preservar la coherencia semántica sin sobrecargar el modelo. Además, esta estrategia es crucial para la indexación efectiva de contenidos en bases de datos vectoriales, lo que mejora la recuperación y relevancia de los resultados en consultas específicas.

Retos de la Chunkerización: A pesar de sus beneficios, la chunkerización presenta varios desafíos. El más significativo es determinar el tamaño óptimo de cada chunk. Si los chunks son demasiado pequeños, pueden perder contexto necesario para una comprensión completa; si son demasiado grandes, pueden exceder las capacidades de procesamiento del modelo y diluir la relevancia del contenido. Otro reto es la selección de los puntos de corte dentro del texto, que debe hacerse de manera que se preserve la integridad del contenido semántico.

La chunkerización es un proceso esencial que requiere un cuidadoso equilibrio entre tamaño de chunk, preservación del contexto y capacidades del modelo. En las siguientes secciones, exploraremos diferentes estrategias y métodos de chunkerización, evaluando sus ventajas y limitaciones en diversos contextos de aplicación.

2.2. Consideraciones sobre la Chunkerización

Al desarrollar una estrategia de chunkerización eficaz, es crucial considerar varios aspectos que influirán en el rendimiento y la eficacia del modelo de lenguaje. A continuación, se detallan algunas de las consideraciones más importantes:

2.2.1. Naturaleza del Contenido

El tipo de contenido que se está indexando es determinante en la selección de la estrategia de chunkerización. Por ejemplo, documentos largos como artículos o libros pueden requerir un enfoque diferente en comparación con contenidos más breves como tuits o mensajes instantáneos. Esta distinción afecta tanto la elección del modelo para crear los embeddings como la estrategia de chunkerización a aplicar.

2.2.2. Modelo y Tamaño Óptimo del Chunk

Dependiendo del modelo utilizado para crear los embeddings, existirán tamaños de chunk en los que el modelo desempeñará mejor. Por ejemplo, algunos modelos están optimizados para trabajar con frases individuales, mientras que otros pueden manejar mejor segmentos de texto más largos. Identificar el tamaño de chunk que maximiza la calidad de los embeddings es crucial para el éxito de la estrategia de chunkerización.

2.2.3. Complejidad y Longitud de las Consultas de Usuario

Las expectativas sobre la longitud y complejidad de las consultas de los usuarios deben guiar la manera en que se chunkeriza el contenido. Si las consultas suelen ser cortas y específicas, es posible que se prefiera chunkerizar el contenido en segmentos más pequeños para reflejar esta especificidad. En cambio, consultas más largas y complejas podrían beneficiarse de chunks más grandes que proporcionen un contexto más amplio.

2.2.4. Uso de los Resultados Recuperados

El propósito final de los chunks recuperados también juega un papel crucial. Dependiendo de si los resultados serán utilizados para búsqueda semántica, respuesta a preguntas, resumen, o cualquier otro fin, la estrategia de chunkerización podría variar para optimizar la relevancia y utilidad de la información recuperada.

2.2.5. Limitaciones Técnicas

Las limitaciones técnicas, como el número máximo de tokens que el modelo puede procesar en una sola instancia o las restricciones de memoria, son críticas para definir el tamaño de los chunks. Estas limitaciones no solo afectan cómo se chunkeriza el contenido sino también cómo se procesa posteriormente en el modelo.

Estas consideraciones son fundamentales para desarrollar una estrategia de chunkerización que no solo sea eficiente sino también efectiva en términos de mejorar la relevancia y precisión de las respuestas del modelo.

2.3. Estrategias de Chunkerización

La chunkerización puede implementarse de diversas maneras, cada una con sus propios beneficios y desafíos. A continuación, exploramos algunas de las estrategias más comunes y cómo pueden optimizarse según diferentes necesidades y contextos.

2.3.1. Chunkerización de Tamaño Fijo

Esta estrategia consiste en dividir el texto en segmentos de un tamaño predeterminado, medido en número de tokens o carácteres. La principal ventaja de este método es su simplicidad y facilidad de implementación, ya que no requiere un análisis profundo del contenido. Sin embargo, un desafío significativo es que puede cortar frases a la mitad, perdiendo contexto o generando chunks que carecen de sentido por sí solos. Para mitigar esto, se puede optar por incluir una superposición entre los chunks, donde el final de un chunk se superpone con el inicio del siguiente, ayudando a preservar el contexto (overload).

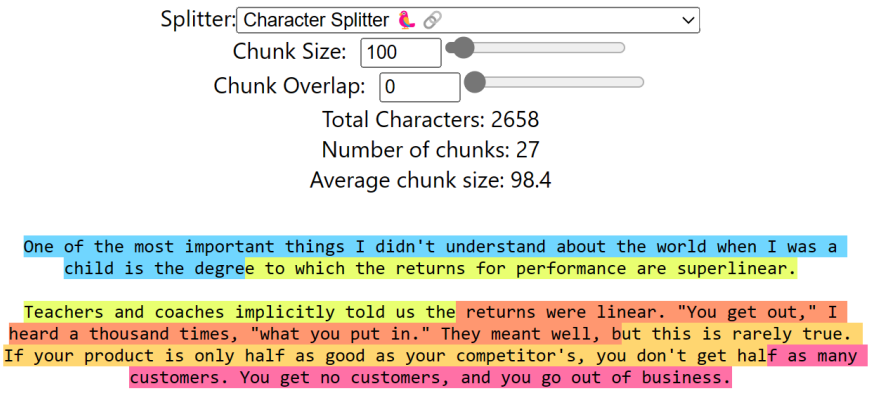

Figura 2.1: *Ejemplo de chunkerización tamaño fijo sin overload*

2.3.2. Chunkerización Consciente del Contenido

A diferencia de la chunkerización de tamaño fijo, este método tiene en cuenta el contenido textual para determinar los puntos de corte. La chunkerización consciente del contenido puede basarse en la detección de fronteras naturales en el texto, como finales de oraciones o párrafos, asegurando que cada segmento contenga una unidad completa de información. Esto ayuda a mantener la coherencia semántica de los chunks, haciendo que cada uno sea significativo por sí mismo y mejore la calidad de las incrustaciones generadas por el modelo. Este método es particularmente útil en documentos con una estructura clara y bien definida, como artículos académicos o reportes técnicos.

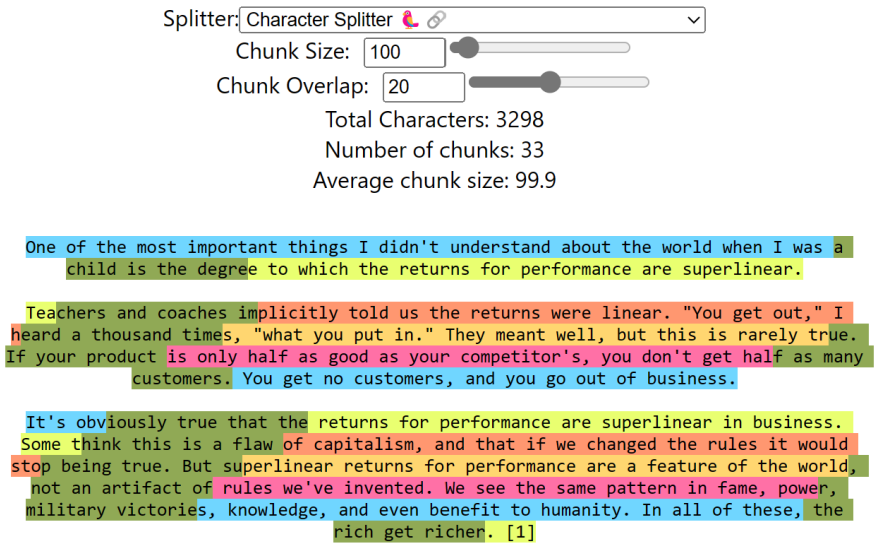

Figura 2.2: *Ejemplo de chunkerización tamaño fijo con overload*

2.3.3. Chunkerización Recursiva

Este enfoque implica un proceso iterativo y jerárquico de dividir el texto en chunks. Inicialmente, el texto se divide utilizando un criterio de separación amplio, y si los chunks resultantes aún son demasiado grandes o no cumplen con ciertos criterios, el proceso se repite en cada chunk hasta alcanzar el tamaño o la estructura deseada. La chunkerización recursiva es útil en textos largos y complejos donde los niveles múltiples de división permiten manejar mejor la diversidad y complejidad del contenido.

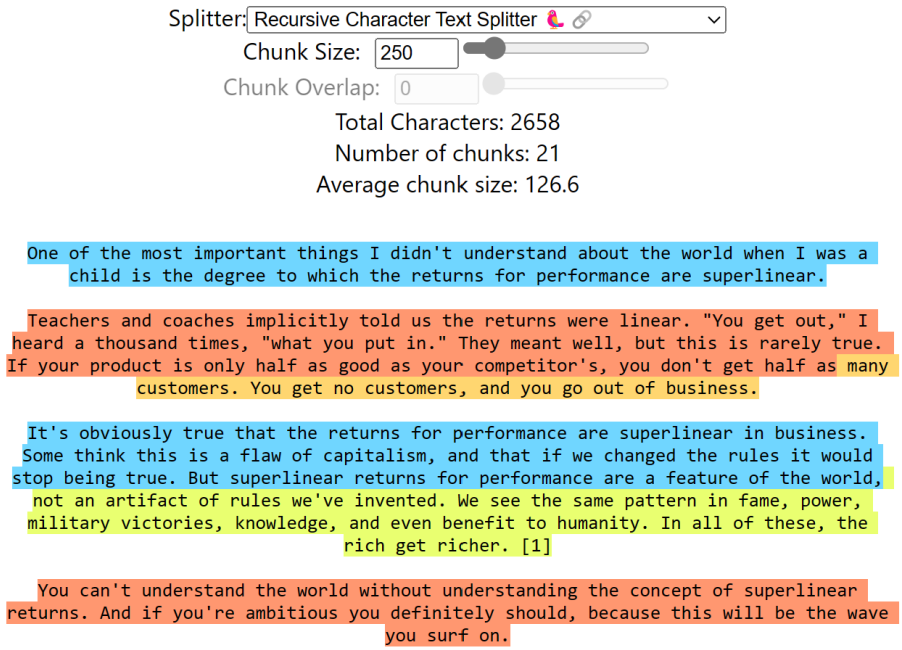

Figura 2.3: *Ejemplo de chunkerización recursiva*

2.3.4. Chunkerización Especializada

Esta estrategia se adapta a formatos de contenido específicos que requieren un tratamiento particular, como puede ser el caso de textos en Markdown o LaTeX. La chunkerización especializada reconoce y respeta la estructura y sintaxis propias de estos formatos, asegurando que los chunks resultantes conserven la integridad y funcionalidad del texto original. Por ejemplo, en un documento LaTeX, los chunks podrían definirse para encapsular secciones completas o subsecciones, preservando las etiquetas y comandos propios del formato.

Cada una de estas estrategias tiene sus propias fortalezas y puede ser más adecuada para diferentes tipos de textos y aplicaciones. La elección de una estrategia de chunkerización debe basarse en una evaluación cuidadosa de los requisitos del proyecto, la naturaleza del contenido y los objetivos específicos del sistema de procesamiento de lenguaje natural.

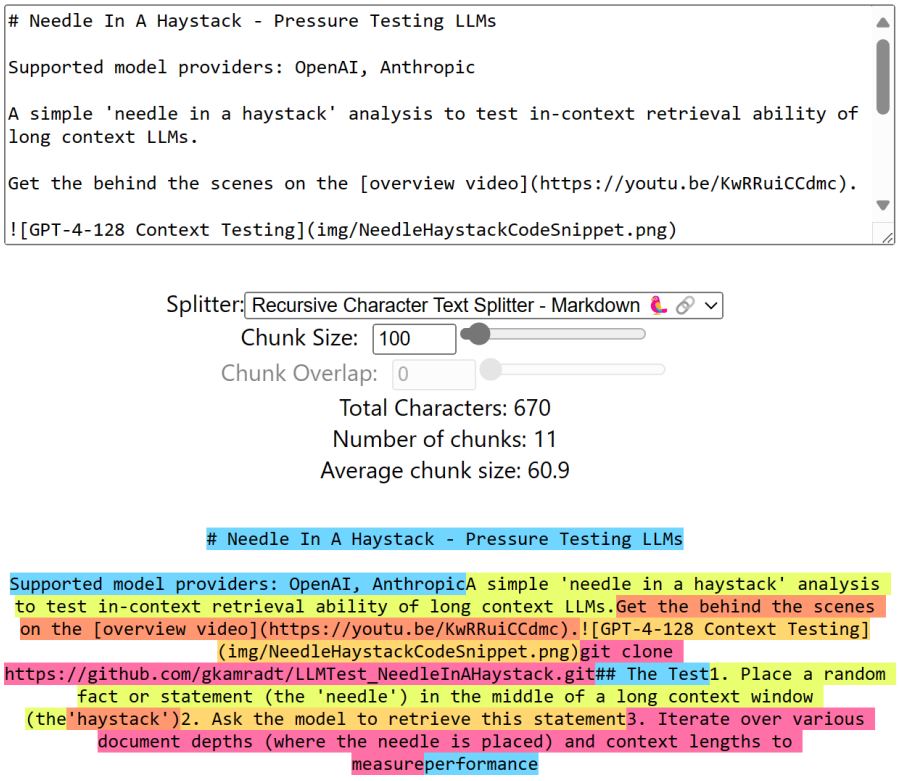

Figura 2.4: *Ejemplo de chunkerización Especializada*

2.3.5. Chunkerización Semántica

La chunkerización semántica es una técnica avanzada que divide el texto en unidades basadas en la relevancia semántica, utilizando modelos de creación de embeddings para evaluar la similitud entre segmentos de texto.

2.3.5.1. Implementación y Ventajas

Esta estrategia se implementa mediante la transformación de texto en representaciones vectoriales y el cálculo de su similitud, agrupando segmentos similares. Es especialmente útil para mantener la coherencia del contenido en tareas de recuperación de información y análisis de texto.

2.3.5.2. Desafíos y Aplicaciones

A pesar de su efectividad, enfrenta desafíos como la alta demanda computacional y la dificultad en determinar umbrales de similitud óptimos. Su aplicación es ideal en sistemas de búsqueda semántica y en el análisis de textos complejos en contextos académicos o de investigación.

2.4. Conclusiones

La chunkerización es fundamental para mejorar la eficiencia y precisión en aplicaciones relacionadas con LLM. No existe una solución única para todos los casos, por lo que es crucial evaluar cada situación individualmente para encontrar la estrategia más efectiva.

Retrievers 3

3.1. Introducción

Habiendo introducido los conceptos fundamentales de los Modelos de Lenguaje a Gran Escala (LLMs) y los RAG en los capítulos anteriores, ahora nos centramos en uno de los componentes críticos que potencian los sistemas RAG: los *retrievers*. Este concepto es esencial para la creación de un RAG sobre un LLM, permitiéndole acceder a información actualizada y relevante más allá de su entrenamiento inicial.

En este capítulo, exploraremos los diferentes tipos de *retrievers*, cómo se integran en los sistemas RAG y las distintas metodologías para su implementación. Discutiremos desde los simples *retrievers* basados en bases de datos vectoriales hasta complejas configuraciones que involucran múltiples métodos de recuperación de información.

3.2. Fundamentos de los Retrievers

3.2.1. Definición y Función

Un *retriever* es una herramienta que busca y recupera información relevante de un conjunto de datos grande y posiblemente no estructurado. En el contexto de RAG, un *retriever* actúa como el puente entre la pregunta del usuario y la base de conocimientos o información almacenada. La eficacia de un *retriever* es crucial, ya que determina la calidad y la relevancia de la información que se utiliza para generar respuestas.

3.2.2. Rol en RAG

Como se puede ver en la figura 3.1, en un sistema RAG, el retriever selecciona fragmentos de texto o documentos que son potencialmente útiles para responder a una consulta específica. Estos documentos se pasan luego a un modelo de lenguaje, que integra esta información para generar una respuesta coherente y contextualmente rica. Esta capacidad de incorporar información dinámica de diversas fuentes externas distingue a los RAGs de los modelos de lenguaje tradicionales que dependen únicamente de lo que han aprendido durante su entrenamiento inicial.

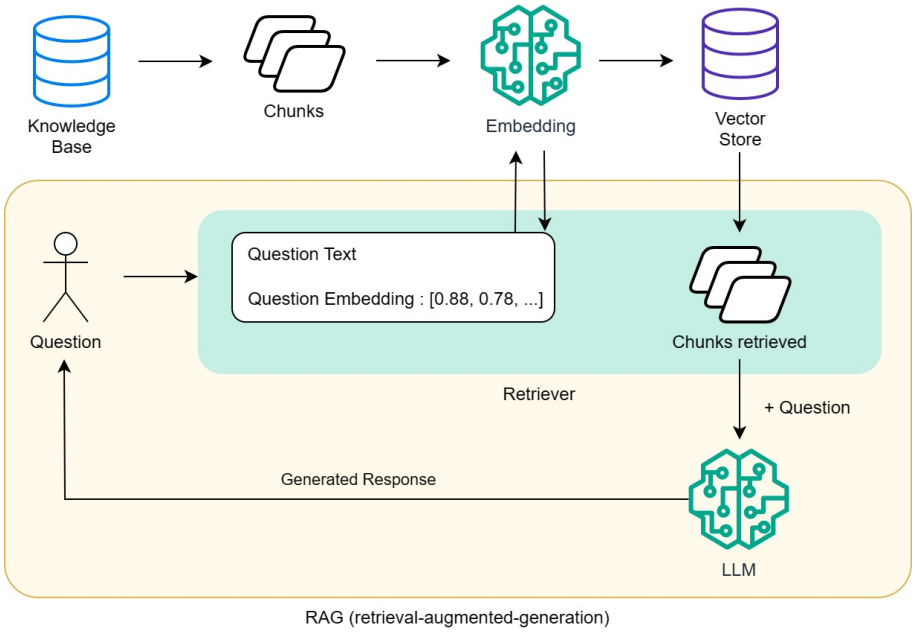

Figura 3.1: *Esquema de un RAG*

3.3. Tipos de Retrievers Tradicionales

Los retrievers se clasifican principalmente en tres tipos, cada uno con sus propias técnicas y aplicaciones preferidas en el procesamiento de consultas y la recuperación de información. Estos son los retrievers *sparse*, *dense* e *hybrid*.

3.3.1. Retrievers Sparse

Los retrievers *sparse* utilizan métodos basados en coincidencia de términos para recuperar documentos. Estos métodos, como TF-IDF (Frequency-Inverse Document Frequency) y BM25, analizan la frecuencia de las palabras en documentos para determinar su relevancia con respecto a una consulta. La principal ventaja de los retrievers sparse es su eficiencia y rapidez, lo que los hace adecuados para grandes volúmenes de datos donde la relevancia se puede medir por la presencia de palabras clave específicas. (Lewis et al., 2020)

Estos retrievers son particularmente útiles en situaciones donde las consultas son directas y las palabras clave bien definidas son suficientes para recuperar información relevante. Sin embargo, pueden no ser efectivos en contextos donde la semántica de la consulta es más importante que las palabras específicas utilizadas.

3.3.2. Retrievers Dense

Por otro lado, los retrievers *dense* emplean modelos de vectorización (embeddings) para representar tanto las consultas como los documentos como vectores densos en un espacio vectorial continuo. Estos modelos capturan la semántica de las palabras y las frases, permitiendo una recuperación más efectiva en casos donde las consultas y los documentos no comparten

términos exactos pero están relacionados en contexto y significado. (Lewis et al., 2020)

Los retrievers dense son esenciales para aplicaciones donde la relevancia semántica entre la consulta y los documentos es más crítica que la coincidencia exacta de palabras. Su capacidad para entender y procesar el lenguaje de manera más natural los hace muy valiosos en sistemas RAG modernos. Estos retrievers son muy dependientes del modelo de vectorización usado.

3.3.3. Retrievers Hybrid

Finalmente, los retrievers *hybrid* combinan las técnicas de los retrievers sparse y dense para aprovechar las ventajas de ambos. Este enfoque permite una recuperación inicial rápida de documentos potencialmente relevantes utilizando métodos sparse, seguida de un refinamiento más detallado y semánticamente rico con técnicas dense.

Los retrievers hybrid ofrecen un equilibrio entre eficiencia y profundidad semántica, lo que los hace ideales para sistemas RAG que requieren tanto precisión en la recuperación de información relevante como la capacidad de manejar grandes volúmenes de datos.

Cada tipo de retriever tiene su lugar en el ecosistema de los RAG, y la elección de uno sobre otro depende en gran medida de las necesidades específicas del sistema y las características de las consultas y los conjuntos de datos utilizados.

3.4. Retrievers Avanzados

Dada la evolución constante de los sistemas RAG y la necesidad de integrar información más específica y contextual, la comunidad ha estado desarrollando diferentes retrievers que tienen sentido en determinados contextos. Estos retrievers utilizan técnicas sofisticadas para mejorar la precisión y la relevancia de la información recuperada. Aquí exploramos varios enfoques que se recopilan en la web de langchain (LangChain, 2024).

3.4.1. Documento Matriz (Parent Document)

Este tipo de retriever es ideal cuando los documentos contienen numerosos fragmentos de información distintos que se benefician de ser indexados individualmente, pero que es preferible recuperar juntos. El proceso involucra la indexación de múltiples fragmentos o trozos de cada documento. Luego, estos fragmentos se buscan en el espacio de vectores (embeddings) para identificar aquellos que son más similares entre sí, pero en lugar de recuperar solo los fragmentos individuales, se recupera y devuelve el documento completo al que pertenecen. Esta metodología asegura que toda la información contextual relacionada esté disponible para generar respuestas más completas y detalladas.

3.4.1.1. Multi Vector

Este retriever es especialmente útil cuando es posible extraer de los documentos información que se considera relevante para indexar (a parte del texto en sí). El proceso implica

la creación de múltiples vectores para cada documento. Cada vector puede ser generado de diversas maneras, incluyendo, por ejemplo, resúmenes del texto, preguntas hipotéticas, descripciones de imágenes o tablas, etc. Esta técnica permite una representación más rica de los documentos, facilitando una recuperación más precisa y alineada con las necesidades específicas del retriever.

3.4.2. Consulta Autónoma (Self Query)

Utiliza un LLM para transformar la entrada del usuario en dos componentes principales: una cadena que se busca semánticamente y un filtro de metadatos que acompaña la consulta. Este método es particularmente valioso porque muchas veces las preguntas están relacionadas con los metadatos de los documentos, no con su contenido textual directo. La capacidad de enfocar la búsqueda en metadatos permite recuperar información que es más relevante para la intención específica del usuario, mejorando así la precisión y utilidad de las respuestas generadas. Este tipo de retriever es especialmente útil cuando las preguntas de los usuarios se responden mejor recuperando documentos basados en metadatos en lugar de en similitudes textuales directas

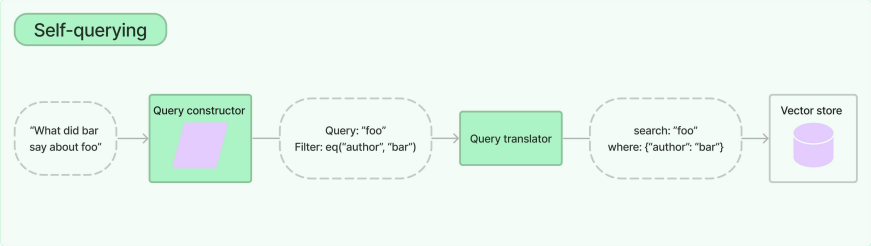

Figura 3.2: *Esquema de un Self Query retriever (LangChain, 2024)*

3.4.2.1. Compresión Contextual

Uno de los desafíos en la recuperación de documentos es que generalmente no se conocen las consultas específicas que enfrentará el sistema de almacenamiento de documentos cuando se ingieren datos en el sistema. Esto significa que la información más relevante para una consulta puede estar enterrada en un documento con mucho texto irrelevante. Pasar ese documento completo a través de la aplicación puede llevar a llamadas más costosas al LLM y a respuestas de menor calidad.

La compresión contextual está diseñada para solucionar esto. En lugar de devolver inmediatamente los documentos recuperados tal como están, se pueden comprimir utilizando el contexto de la consulta dada, de modo que solo se devuelva la información relevante.

3.4.3. Time-Weighted Vectorstore

Optimiza la recuperación basándose tanto en la similitud semántica como en la fechateams del documento. Es ideal para aplicaciones donde la información más actual es crucial, como

en el seguimiento de noticias o tendencias del mercado.

3.4.4. Multi-Query Retriever

Genera múltiples consultas a partir de una inicial para abordar consultas complejas que requieren información sobre varios temas. Luego, recupera documentos para cada una de estas consultas, lo que garantiza una respuesta exhaustiva y detallada. Es muy útil cuando las preguntas van a tratar sobre varios temas y temas no relacionados.

3.4.5. Long-Context Reorder

Es especialmente útil en modelos que enfrentan degradaciones de rendimiento cuando deben acceder a información relevante en medio de contextos extensos. Este tipo de retriever reordena los documentos recuperados para optimizar la atención del modelo a la información más pertinente, colocando los documentos más relevantes al principio y al final, mientras que los menos relevantes quedan en el medio. El "Long Context Reorderütiliza un enfoque específico conocido como "Lost in the middle"para contrarrestar el problema de que los modelos ignoren documentos importantes simplemente porque aparecen en posiciones menos destacadas dentro de un conjunto de datos grande. (Liu et al., 2023)

Estos retrievers avanzados representan un conjunto diverso de herramientas diseñadas para mejorar la eficiencia y efectividad de los sistemas RAG, facilitando respuestas más precisas y contextuales en una amplia variedad de aplicaciones.

3.5. Conclusión

A lo largo de este capítulo, hemos explorado en profundidad los diversos tipos de *retrievers* que potencian los sistemas de RAG. Hemos visto cómo los *retrievers* no solo facilitan la recuperación de información relevante y actualizada, sino que también optimizan la eficiencia del procesamiento y la precisión de las respuestas generadas por los LLMs. La implementación adecuada de estos *retrievers* puede marcar una diferencia significativa en la capacidad de un sistema RAG para proporcionar respuestas precisas, contextuales y de alta calidad.

Este capítulo también ha subrayado la importancia de elegir el tipo correcto de *retriever* según las necesidades específicas del sistema y del dominio de aplicación. Con las tecnologías emergentes y los desarrollos continuos en el campo de la inteligencia artificial, es probable que veamos aún más innovaciones en los métodos de recuperación, lo que a su vez podría ampliar aún más las capacidades y aplicaciones de los RAGs.

En resumen, los *retrievers* son componentes cruciales que no solo enriquecen la funcionalidad de los sistemas basados en LLMs mediante la incorporación de conocimientos actualizados y contextualmente relevantes, sino que también representan un área de investigación activa y en evolución que continuará influyendo en el futuro de la generación de lenguaje y la recuperación de información.

Lista de Acrónimos

Apéndize A

A

Apéndize B

B

Bibliografía

LangChain (2024). Documentation for retriever modules.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., y Liang, P. (2023). Lost in the middle: How language models use long contexts.