Apuntes semana 12 - Modelos de Lenguaje Extensos y Sistemas Avanzados (LLMs, RAG y Agentes Inteligentes)

Fernando Daniel Brenes Reyes
Escuela de Ingeniería en Computación
Instituto Tecnológico de Costa Rica
Cartago, Costa Rica
21 de octubre
2020097446@estudiantec.cr

***Resumen*—El presente documento contiene un repaso y ampliación de los conceptos fundamentales de los Modelos de Lenguaje Extensos (LLMs), su representación del conocimiento mediante la tokenización y los embeddings en espacios vectoriales. Se detalla la evolución del LLM tradicional hacia arquitecturas avanzadas como Retrieval-Augmented Generation (RAG), que resuelve las limitaciones de conocimiento estático, y los Agentes Inteligentes, que integran memoria, planificación y la capacidad de ejecutar acciones autónomas, reflejando el estado del arte en la inteligencia artificial contextual y adaptable.**

***Index Terms*—LLM, RAG, Agentes Inteligentes, Tokenización, Embeddings, Aprendizaje Contextual.**

I. Introducción

Los **Modelos de Lenguaje Extensos (LLMs)** se han consolidado como la base de los sistemas modernos de **Inteligencia Artificial Generativa (IAG)**. Estos modelos no solo generan texto, sino que también permiten la comprensión y el razonamiento sobre texto, código y otra información compleja. Aunque son potentes, los LLMs poseen un conocimiento limitado a sus datos de entrenamiento (**estático**) y pueden incurrir en **alucinaciones**. Para superar estas barreras, se han desarrollado enfoques como Retrieval-Augmented Generation (RAG) y los Agentes Inteligentes.

II. Fundamentos de LLMs y Representación

II-A. Tokenización: De la Palabra al Número

Para que los LLMs puedan computar con el lenguaje, el texto de entrada debe convertirse en una representación numérica. El proceso de **Tokenización** transforma palabras, signos o símbolos en unidades mínimas llamadas **tokens**, asignando a cada una un **ID numérico** único.

Existen múltiples estrategias de tokenización, cada una optimizada para un objetivo distinto:

- **Por palabra**: Ofrece simplicidad.
- **Por carácter**: Permite manejar símbolos o palabras fuera del vocabulario (OOV).
- **Subpalabra (BPE, WordPiece)**: Logra un equilibrio óptimo entre el tamaño del vocabulario y la preservación del contexto.

II-B. Embeddings y Espacios Vectoriales

Una vez tokenizados, los IDs numéricos se convierten en **embeddings**, que son representaciones numéricas densas en un espacio continuo de alta dimensión.

- **Captura semántica**: Los embeddings capturan el significado y las relaciones contextuales entre palabras u oraciones completas.
- **Proximidad**: Las palabras con significados similares se ubican próximas en el espacio vectorial.
- **Operaciones**: Este espacio permite realizar operaciones semánticas, como analogías (por ejemplo, $Rey - Hombre + Mujer \approx Reina$).

Para medir la **similitud** entre dos vectores $\mathbf{a}$ y $\mathbf{b}$ en $\mathbb{R}^n$, la **Similitud del Coseno** es la métrica más utilizada:

$$sim(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||||\mathbf{b}||} \tag{1}$$

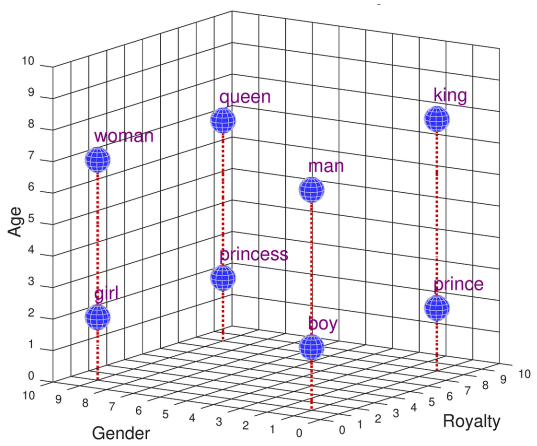

Figura 1. Representación tridimensional de tokens (Realeza).

II-C. *Capacidades Emergentes*

El entrenamiento masivo de los LLMs les confiere capacidades avanzadas que **emergen** sin haber sido entrenados directamente para ellas:

- **Razonamiento y planificación.**
- **Aprendizaje en el prompt** (*In-context Learning*): Adaptan el comportamiento a partir de ejemplos dados en la entrada.
- **Multitarea**: Realizan traducción, clasificación y codificación sin reentrenamiento.

III. Retrieval-Augmented Generation (RAG)

RAG es un paradigma que conecta un LLM con un módulo de recuperación (*retriever*) para inyectar **conocimiento externo, actualizado y verificable** durante la generación de respuestas.

III-A. *Proceso y Flujo de RAG*

1. **Preparación (Chunking)**: Los documentos se dividen en fragmentos (**chunks**), que suelen contener entre **200 y 500 tokens**, a menudo con *overlap* para preservar el contexto.
2. **Indexación**: Cada *chunk* se convierte en un **embedding** y se almacena en una **base de datos vectorial** (por ejemplo, FAISS, Qdrant, Pinecone).
3. **Consulta y recuperación**: La pregunta del usuario se transforma en un *embedding*, se calcula la similitud con los vectores indexados y se seleccionan los **top-k chunks** más cercanos semánticamente.
4. **Aumento y generación**: Los *chunks* recuperados se integran en una plantilla estructurada (*prompt*) como **contexto adicional**, asegurando que la respuesta del LLM sea precisa y fundamentada.

III-B. *Ventajas y Limitaciones*

RAG ofrece la **reducción de alucinaciones**, la **actualización continua del conocimiento** y la aplicabilidad en dominios especializados. No obstante, los sistemas RAG siguen siendo **pasivos**; su función se limita a complementar la respuesta del LLM con datos recuperados.

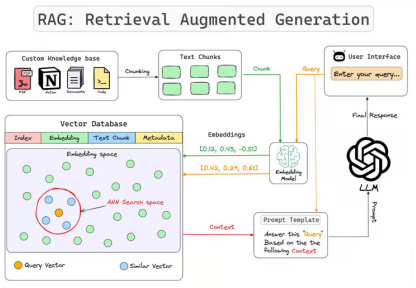

Figura 2. Diagrama del flujo de un sistema RAG, desde la indexación hasta la generación de la respuesta.

Figura 3. Agente inteligente.

IV. De LLM a Agente Inteligente

Los **Agentes Inteligentes** basados en LLMs superan la pasividad de los sistemas RAG. Estos agentes pueden **razonar, planificar** y **actuar** de manera autónoma, interactuando con el mundo real mediante herramientas externas.

IV-A. *Componentes Clave del Agente*

1. **Memoria**: Permite mantener coherencia y contexto a lo largo del tiempo.
   - **Corto plazo**: Ventana de contexto del modelo.
   - **Largo plazo**: Bases de datos externas, incluyendo sistemas RAG para la recuperación contextual.
2. **Planificación**: Permite descomponer problemas complejos en pasos y razonar sobre ellos.
   - **Chains of Thought (CoT)**: Razonamiento secuencial.
   - **Trees of Thought (ToT)**: Exploración de múltiples caminos de razonamiento antes de decidir.
3. **Acción**: Capacidad de ejecutar tareas concretas mediante **herramientas externas** (APIs, buscadores, sistemas RAG). Por ejemplo, un agente puede acceder a un sistema de recursos humanos para responder: "¿Cuántos días de vacaciones me quedan?".

IV-B. *Escalamiento Responsable*

La implementación de agentes requiere evaluar cuándo es necesaria la complejidad de un sistema multiagente. Es crucial garantizar la **seguridad**, **privacidad** y el **uso ético** de los datos, diseñando los agentes bajo principios de transparencia y responsabilidad.

Referencias

[1] Pacheco Portuguez, S. (2025). *Presentación del curso de Inteligencia Artificial*. Instituto Tecnológico de Costa Rica.