Prompting: RAG, Chain of Thought y otros

Julia Milanese
juliamilanese@gmail.com

Clase 3 Parte 2
Martes 29/04/2025

- Clase 1: Nociones básicas de redes neuronales para el procesamiento del lenguaje natural
  - Parte 1: Tokenización, embeddings estáticos
  - Parte 2: el perceptrón; arquitectura básica de redes neuronales; funciones de activación; backpropagation.
- Clase 2: Grandes y pequeños modelos de lenguaje.
  - Parte 1: Mecanismo de atención; transformers; embeddings posicionales
  - Parte 2: Similitudes y diferencias entre modelos de lenguaje
- Clase 3: ***Prompt engineering***
  - Parte 1: Hiperparámetros de los modelos de lenguaje; prompting: Zero-Shot; Few-Shot
  - **Parte 2: Prompting: RAG, Chain of Thought y otros**

Presentación

Estructura y temas de la clase de hoy:

1. Introducción
2. Prompting
3. Question Answering
4. Limitaciones de los LLMs
5. Information Retrieval (IR)
6. Retrieval-Augmented Generation (RAG)
7. CoT
8. Hacia los agentes inteligentes
9. ReAct Prompting
10. Elementos de la práctica de hoy
11. Recapitulación
12. Bibliografía

Prompt engineering

Definición:
El *prompt engineering* es la práctica de diseñar y optimizar entradas (*prompts*) para modelos de lenguaje con el fin de guiar sus respuestas hacia un comportamiento deseado o una salida específica.

Prompt engineering

Definición:
El *prompt engineering* es la práctica de diseñar y optimizar entradas (*prompts*) para modelos de lenguaje con el fin de guiar sus respuestas hacia un comportamiento deseado o una salida específica.
Es una técnica esencial cuando se interactúa con modelos como *ChatGPT*, *GPT-4*, *Claude*, *LLaMA*, entre otros.

Características del Prompt Engineering (1/3)

- **Especificidad del lenguaje:** usar lenguaje claro y preciso. Pequeños cambios pueden alterar los resultados.

Características del Prompt Engineering (1/3)

- **Especificidad del lenguaje:** usar lenguaje claro y preciso. Pequeños cambios pueden alterar los resultados.
- **Contextualización:** incluir ejemplos, roles o contexto mejora la respuesta (e.g., “Actúa como un asistente...").

Características del Prompt Engineering (1/3)

- **Especificidad del lenguaje:** usar lenguaje claro y preciso. Pequeños cambios pueden alterar los resultados.
- **Contextualización:** incluir ejemplos, roles o contexto mejora la respuesta (e.g., “Actúa como un asistente...").
- **Formatos comunes:** preguntas abiertas/cerradas, instrucciones paso a paso, *few-shot* y *zero-shot*.

Características del Prompt Engineering (2/3)

- **Iteración y prueba:** diseño iterativo basado en prueba y error.

Características del Prompt Engineering (2/3)

- **Iteración y prueba:** diseño iterativo basado en prueba y error.
- **Optimización por tarea:** diferentes estrategias según la tarea (resumen, generación, clasificación...).

Características del Prompt Engineering (2/3)

- **Iteración y prueba:** diseño iterativo basado en prueba y error.
- **Optimización por tarea:** diferentes estrategias según la tarea (resumen, generación, clasificación...).
- **Transferencia entre modelos:** algunos *prompts* se adaptan bien, otros requieren ajustes.

Características del Prompt Engineering (3/3)

- **Longitud del prompt:** prompts muy largos pueden ser truncados o dar resultados erróneos.

Características del Prompt Engineering (3/3)

- **Longitud del prompt:** prompts muy largos pueden ser truncados o dar resultados erróneos.
- **Dependencia del modelo:** cada modelo puede responder de forma distinta al mismo *prompt*.

Características del Prompt Engineering (3/3)

- **Longitud del prompt:** prompts muy largos pueden ser truncados o dar resultados erróneos.
- **Dependencia del modelo:** cada modelo puede responder de forma distinta al mismo *prompt*.
- **Formatos estructurados:** uso de listas, tablas, JSON o Markdown como parte del *prompt*.

Características del Prompt Engineering (3/3)

- **Longitud del prompt:** prompts muy largos pueden ser truncados o dar resultados erróneos.
- **Dependencia del modelo:** cada modelo puede responder de forma distinta al mismo *prompt*.
- **Formatos estructurados:** uso de listas, tablas, JSON o Markdown como parte del *prompt*.
- **Estrategias avanzadas:** *Chain-of-Thought (CoT)*, *ReAct*, *RAG*.

La Necesidad del Question Answering

- *El QA responde a una necesidad humana básica: saber cosas.* Jurafsky y Martin (2025)
- Desde los primeros sistemas computacionales, se intentó obtener respuestas automáticas.
- Evolución: de motores de búsqueda tradicionales a LLMs modernos.
- Las *factoid questions* representan una categoría útil y concreta de QA.
  - ¿Quién escribió El Quijote? vs ¿Por qué es importante la fotosíntesis?

Desafíos del QA con Prompting Simple

- **Alucinaciones**: Respuestas convincentes pero falsas.
- **Descalibración**: Alta confianza en respuestas incorrectas.
- **Acceso a Datos Privados**: No es posible sin integración externa.
- **Información Actualizada**: Los LLMs estáticos no pueden responder sobre eventos recientes.

¿Qué es Information Retrieval?

¿Qué es Information Retrieval?

- IR es el campo dedicado a recuperar información relevante según la necesidad del usuario.

¿Qué es Information Retrieval?

- IR es el campo dedicado a recuperar información relevante según la necesidad del usuario.
- Arquitectura típica: procesamiento de la consulta, indexación, búsqueda.

¿Qué es Information Retrieval?

- IR es el campo dedicado a recuperar información relevante según la necesidad del usuario.
- Arquitectura típica: procesamiento de la consulta, indexación, búsqueda.
- Aplicación principal: motores de búsqueda.

¿Qué es RAG?

Retrieval-Augmented Generation (RAG) es una técnica que combina modelos generativos de lenguaje con recuperación de información para responder preguntas de forma precisa y fundamentada.

¿Qué es RAG?

Retrieval-Augmented Generation (RAG) es una técnica que combina modelos generativos de lenguaje con recuperación de información para responder preguntas de forma precisa y fundamentada.

- Utiliza una base de conocimiento externa para extraer información relevante.

¿Qué es RAG?

Retrieval-Augmented Generation (RAG) es una técnica que combina modelos generativos de lenguaje con recuperación de información para responder preguntas de forma precisa y fundamentada.

- Utiliza una base de conocimiento externa para extraer información relevante.
- Mejora la precisión y fiabilidad, reduciendo alucinaciones típicas de modelos generativos puros.

¿Qué es RAG?

Retrieval-Augmented Generation (RAG) es una técnica que combina modelos generativos de lenguaje con recuperación de información para responder preguntas de forma precisa y fundamentada.

- Utiliza una base de conocimiento externa para extraer información relevante.
- Mejora la precisión y fiabilidad, reduciendo alucinaciones típicas de modelos generativos puros.
- Es especialmente útil para dominios específicos y preguntas sobre datos privados o eventos recientes.

Funcionamiento de RAG

RAG opera en dos fases:

Funcionamiento de RAG

RAG opera en dos fases:

1. **Recuperación (Retrieval)**:

Funcionamiento de RAG

RAG opera en dos fases:

1. **Recuperación (Retrieval)**:
   - La consulta se convierte en un vector semántico (embedding).
   - Se recuperan documentos relevantes mediante búsqueda vectorial o semántica.

Funcionamiento de RAG

RAG opera en dos fases:

1. **Recuperación (Retrieval)**:
   - La consulta se convierte en un vector semántico (embedding).
   - Se recuperan documentos relevantes mediante búsqueda vectorial o semántica.
2. **Generación (Generation)**:
   - Un modelo generativo produce la respuesta utilizando el contexto recuperado.

Funcionamiento de RAG

RAG opera en dos fases:

1. **Recuperación (Retrieval)**:
   - La consulta se convierte en un vector semántico (embedding).
   - Se recuperan documentos relevantes mediante búsqueda vectorial o semántica.
2. **Generación (Generation)**:
   - Un modelo generativo produce la respuesta utilizando el contexto recuperado.

Este enfoque garantiza respuestas ancladas en hechos verificables.

Chain of Thoughts

Chung *et al.* (2022) Finetuning language models on a collection of datasets phrased as instructions has been shown to improve model performance and generalization to unseen tasks.

El **Chain of Thought (CoT) prompting** extiende el *few-shot prompting* al incluir no solo la entrada y la salida, sino también una **cadena de pensamiento**.

Chain of Thoughts

Chung *et al.* (2022) Finetuning language models on a collection of datasets phrased as instructions has been shown to improve model performance and generalization to unseen tasks.

El **Chain of Thought (CoT) prompting** extiende el *few-shot prompting* al incluir no solo la entrada y la salida, sino también una **cadena de pensamiento**.
Esta **cadena de pensamiento** consiste en una secuencia de pasos intermedios, en lenguaje natural, que guían desde la pregunta inicial hasta la respuesta final.

Chain of Thoughts

Chung *et al.* (2022) Finetuning language models on a collection of datasets phrased as instructions has been shown to improve model performance and generalization to unseen tasks.

El **Chain of Thought (CoT) prompting** extiende el *few-shot prompting* al incluir no solo la entrada y la salida, sino también una **cadena de pensamiento**.
Esta **cadena de pensamiento** consiste en una secuencia de pasos intermedios, en lenguaje natural, que guían desde la pregunta inicial hasta la respuesta final.
Cada ejemplo en un *prompt* de CoT incluye:

- **Entrada (Input):** la pregunta o el problema.

Chain of Thoughts

Chung *et al.* (2022) Finetuning language models on a collection of datasets phrased as instructions has been shown to improve model performance and generalization to unseen tasks.

El **Chain of Thought (CoT) prompting** extiende el *few-shot prompting* al incluir no solo la entrada y la salida, sino también una **cadena de pensamiento**.
Esta **cadena de pensamiento** consiste en una secuencia de pasos intermedios, en lenguaje natural, que guían desde la pregunta inicial hasta la respuesta final.
Cada ejemplo en un *prompt* de CoT incluye:

- **Entrada (Input):** la pregunta o el problema.
- **Cadena de Pensamiento (Chain of Thought):** pasos de razonamiento explicativos.

Chain of Thoughts

Chung *et al.* (2022) Finetuning language models on a collection of datasets phrased as instructions has been shown to improve model performance and generalization to unseen tasks.

El **Chain of Thought (CoT) prompting** extiende el *few-shot prompting* al incluir no solo la entrada y la salida, sino también una **cadena de pensamiento**.
Esta **cadena de pensamiento** consiste en una secuencia de pasos intermedios, en lenguaje natural, que guían desde la pregunta inicial hasta la respuesta final.
Cada ejemplo en un *prompt* de CoT incluye:

- **Entrada (Input):** la pregunta o el problema.
- **Cadena de Pensamiento (Chain of Thought):** pasos de razonamiento explicativos.
- **Salida (Output):** la respuesta final.

¿Por qué usar Chain of Thought?

La idea detrás de esta técnica es **imitar el proceso de pensamiento humano**.

¿Por qué usar Chain of Thought?

La idea detrás de esta técnica es **imitar el proceso de pensamiento humano**.
Al ver ejemplos de cómo se descompone un problema complejo en pasos más pequeños y se razona a través de ellos, **el modelo aprende** a generar su propia cadena de pensamiento.

¿Por qué usar Chain of Thought?

La idea detrás de esta técnica es **imitar el proceso de pensamiento humano**.
Al ver ejemplos de cómo se descompone un problema complejo en pasos más pequeños y se razona a través de ellos, **el modelo aprende** a generar su propia cadena de pensamiento.
Esto hace que la resolución sea más lógica y comprensible.

Ventaja principal de CoT

Gracias a este enfoque, el modelo puede:

- Abordar problemas complejos **paso a paso**.

Ventaja principal de CoT

Gracias a este enfoque, el modelo puede:

- Abordar problemas complejos **paso a paso**.
- Desarrollar un **razonamiento más estructurado**.

Ventaja principal de CoT

Gracias a este enfoque, el modelo puede:

- Abordar problemas complejos **paso a paso**.
- Desarrollar un **razonamiento más estructurado**.
- Llegar a soluciones con mayor **coherencia y precisión**.

Ventaja principal de CoT

Gracias a este enfoque, el modelo puede:

- Abordar problemas complejos **paso a paso**.
- Desarrollar un **razonamiento más estructurado**.
- Llegar a soluciones con mayor **coherencia y precisión**.
- Imitar cómo una persona **piensa antes de responder**.

Ventaja principal de CoT

Gracias a este enfoque, el modelo puede:

- Abordar problemas complejos **paso a paso**.
- Desarrollar un **razonamiento más estructurado**.
- Llegar a soluciones con mayor **coherencia y precisión**.
- Imitar cómo una persona **piensa antes de responder**.
- Presenta inconsistencias en razonamientos lógicos complejos y no tan complejos.

Hacia los agentes inteligentes

Russell y Norvig (1995) describen a un agente del siguiente modo:

- Un agente es cualquier cosa que puede percibir su entorno a través de sensores y actuar sobre ese entorno a través de actuadores.
- La función del agente especifica la acción que un agente toma en respuesta a cualquier secuencia de percepciones.

¿Qué es un agente inteligente basado en LLMs?

- Un agente inteligente basado en LLMs es un agente que utiliza un modelo de lenguaje para tomar decisiones y realizar acciones en un entorno.
- El entorno puede ser cualquier cosa que pueda ser percibido por el agente, generalmente una API.
- Un API es una interfaz de programación de aplicaciones que permite a los programas comunicarse entre sí.

En el texto de Yao *et al.* (2023) introducen el ReAct Prompting como parte de un trabajo de pasantía en Google Research. Proponen este tipo de interacción con modelos de lenguaje combinando dos estrategias que ya existían: CoT y Acting. Según los autores:

- La inteligencia humana combina de forma natural el razonamiento verbal con acciones orientadas a tareas.
- Sin embargo, el razonamiento (e.g., chain-of-thought) y la acción (e.g., generación de planes de acción) se han estudiado principalmente por separado.
- Existe la posibilidad de combinar el razonamiento verbal con la toma de decisiones interactiva en sistemas autónomos.
- Así surge Re (Reasinigng) + Act (Acting) Prompting, un enfoque de prompting que combina razonamiento y acción.

ReAct: Razonamiento + Acción

- ReAct solicita a los LLMs que generen tanto razonamientos verbales como acciones relacionadas con una tarea de forma intercalada.
- Esto permite al modelo realizar un razonamiento dinámico para crear, mantener y ajustar planes de acción de alto nivel (razonar para actuar).
- También permite interactuar con entornos externos (e.g., Wikipedia) para incorporar información adicional al razonamiento (actuar para razonar).

- El espacio de acción del agente se aumenta a $\hat{A} = A \cup L$, donde L es el espacio del lenguaje.
- Las acciones en el espacio del lenguaje ($\hat{a}_t \in L$) se refieren a pensamientos o rastros de razonamiento que no afectan al entorno externo.
- Los pensamientos ayudan a componer información útil razonando sobre el contexto actual c_t y actualizan el contexto a $c_{t+} = (c_t, \hat{a}_t)$ para apoyar el razonamiento o la acción futuros.

¿Qué es una Trayectoria en ReAct?

- En el contexto de ReAct, una **trayectoria** es la secuencia de:
  - **Pensamientos (Thoughts)**: Razonamiento verbal del modelo.
  - **Acciones (Actions)**: Interacciones con el entorno o finalización de la tarea.
  - **Observaciones (Observations)**: Información recibida del entorno después de una acción.
- La trayectoria representa el **proceso de resolución de problemas** del agente.
- Los pensamientos y las acciones se generan de forma **intercalada**.

Componentes de una Trayectoria ReAct

- **Pensamiento (Thought)**:
  - Razonamiento en lenguaje natural para guiar la acción.
  - Puede incluir descomposición de objetivos, planificación, razonamiento de sentido común, etc..
- **Acción (Action)**:
  - Interacción con el entorno (e.g., búsqueda en Wikipedia con `search`, `lookup`).
  - Finalización de la tarea con una respuesta (`finish`).
- **Observación (Observation)**:
  - Respuesta del entorno a la acción realizada (e.g., resultados de búsqueda).

Ejemplo Simplificado de Trayectoria ReAct (HotpotQA)

Pregunta: ¿Cuál es el rango de elevación del sector este de la orogenia de Colorado?

- **Pensamiento 1**: Necesito buscar .ºrogenia de Coloradoz encontrar el rango de elevación del sector este.
- **Acción 1**: Buscar[orogenia de Colorado]
- **Observación 1**: La orogenia de Colorado fue un episodio de formación de montañas...
- **Pensamiento 2**: No menciona el sector este. Necesito buscar "sector este".
- **Acción 2**: Buscar[sector este] (o `lookup`)

- **Observación 2**: El sector este se extiende hacia las High Plains...
- **Pensamiento 3**:... necesito buscar "High Plainsz encontrar su rango de elevación.
- **Acción 3**: Buscar[High Plains (Estados Unidos)]
- **Observación 4**: Las High Plains se elevan en elevación desde alrededor de 1,800 a 7,000 pies...
- **Pensamiento 5**: Las High Plains se elevan de 1,800 a 7,000 pies, así que la respuesta es 1,800 a 7,000 pies.
- **Acción 5**: Finalizar[1,800 a 7,000 pies]

Importancia de las Trayectorias en ReAct

- **Interpretabilidad**: Las trayectorias permiten a los humanos comprender el proceso de razonamiento y toma de decisiones del modelo.
- **Diagnóstico**: Facilita la identificación de errores y modos de fallo en el comportamiento del agente.
- **Controlabilidad**: La capacidad de inspeccionar y potencialmente editar los pensamientos en la trayectoria abre la puerta a la corrección del comportamiento en bucle con humanos.
- Las trayectorias de ReAct son más **fundamentadas en hechos** y **confiables** en comparación con el razonamiento aislado.

¿Qué es una API?

API = Interfaz de Programación de Aplicaciones Una API permite que diferentes programas se comuniquen de forma estandarizada. Por ejemplo: una app del clima puede preguntar a un servidor: *"¿Qué clima hace ahora en París?"*

¿Qué es una clave API (API Key)?

Es como una llave o pase privado

- Identifica quién está usando la API.
- Permite controlar cuántas veces la usas.
- Protege el sistema del mal uso.

Ejemplo: Para usar la API de OpenAI (como ChatGPT), necesitas tu propia clave: https://platform.openai.com/api-keys

¿Qué es LangChain?

LangChain = Librería para crear apps con IA

- LangChain es un framework de código abierto diseñado para simplificar la creación de aplicaciones que usan modelos de lenguaje grandes.
- Ofrece una interfaz estándar para construir cadenas de tareas, muchas integraciones con otras herramientas y cadenas completas listas para usar en aplicaciones comunes.
- LangChain permite a los desarrolladores crear aplicaciones que combinan modelos de lenguaje con fuentes externas de datos y herramientas de procesamiento.

LangChain ayuda a los desarrolladores a construir cosas útiles con inteligencia artificial.

¿Qué es LangGraph?

LangGraph = Framework de código abierto para agentes de inteligencia artificial

- Está diseñado para construir, desplegar y gestionar flujos de trabajo complejos con agentes de IA generativa.
- Ofrece un conjunto de herramientas y librerías que permiten crear, ejecutar y optimizar modelos de lenguaje de forma escalable y eficiente.
- En su esencia, LangGraph utiliza arquitecturas basadas en grafos para representar y controlar las relaciones entre los distintos componentes que forman parte del flujo de trabajo de un agente de IA.

En esta clase introdujimos los siguientes temas:

- Prompt Engineering
- RAG
- CoT
- Utilización de herramientas en línea para interactuar con modelos
- Agentes inteligentes.
- ReAct prompting.
- Uso de APIs.
- LangChain y LangGraph

Bibliografía I

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., y Wei, J. (2022). Scaling instruction-finetuned language models.

Jurafsky, D. y Martin, J. H. (2025). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3 edición. Online manuscript released January 12, 2025.

Russell, S. J. y Norvig, P. (1995). *Artificial intelligence: a modern approach*. Pearson Education Limited, Malaysia.

Bibliografía II

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., y Cao, Y. (2023). React: Synergizing reasoning and acting in language models. *ICLR 2023*.