Redes Neuronales para Lenguaje Natural

2024

Grupo de Procesamiento de Lenguaje Natural
Instituto de Computación


Prompting

Prompting

Una **prompt** es una cadena de texto (string) que se ingresa a un modelo de lenguaje para hacer que el modelo resuelva una tarea particular.

Dada una tarea a resolver, se siguen los pasos:

1. Se crea una **plantilla específica** de la tarea con parámetro(s) libre(s) para el texto de entrada.
2. Dado el texto de entrada, se **instancia la prompt** para el LLM.
3. Se realiza **decodificación autorregresiva** para generar una secuencia de tokens de salida.
4. La salida puede ser la deseada o puede ser necesario **extraer la respuesta final** (postprocesamiento).

Ejemplos de plantillas (templates)

Resumen Automático {input} tl-dr:
Traducción Automática {input} Traducir al inglés:
Análisis de sentimiento {input} El sentimiento predominante en el texto anterior es:
Análisis de sentimiento (otra opción) ### Texto: {input} ### Pregunta: El texto anterior ¿Tiene sentimiento positivo o negativo? Opciones: (a) Positivo (b) Negativo ### Respuesta: Creo que la opción correcta es (


In-Context Learning

In-Context Learning

Preentrenamiento:

- Tarea de modelado de lenguaje
- Se ajustan los pesos
- Dataset sin anotar con miles de millones de tokens

Fine-tuning:

- Adaptar modelo a una tarea particular
- Se ajustan los pesos
- Dataset anotado con miles de ejemplos

In-context learning:

- Adaptar modelo a una tarea particular
- No se ajustan los pesos
- Instrucciones de la tarea y (opcionalmente) decenas de ejemplos

Few-Shot Learning

Zero-Shot

Traducir de español a inglés:
queso =>

Instrucción de la tarea

Instancia a resolver

Few-Shot

Traducir de español a inglés:
menta => mint
lobo de mar => sea dog
jirafa de peluche => plush giraffe
queso =>

Instrucción de la tarea

Ejemplos few-shot

Instancia a resolver

Language Models are Few-Shot Learners
Brown et al., (2020)

Few-Shot Learning

¿Cuántos ejemplos?

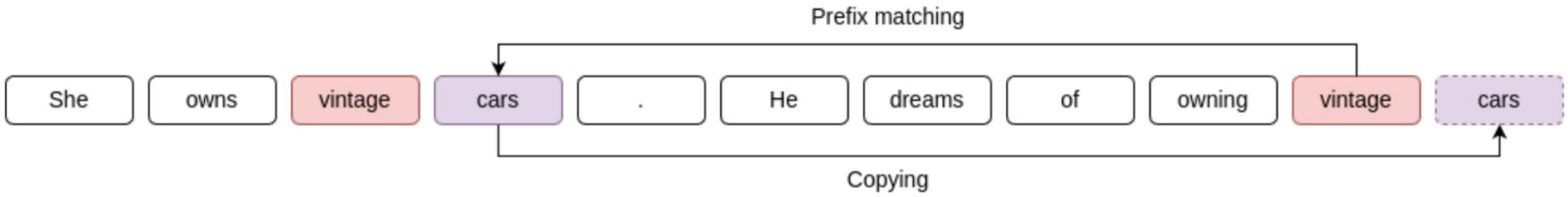
- Una cantidad chica de ejemplos aleatorios es suficiente para mejorar zero-shot.
- Las mayores ganancias tienden a venir del primer ejemplo.
- El beneficio principal es demostrar la tarea que se debe realizar y el formato esperado.
- Incluso ejemplos con respuestas incorrectas pueden mejorar el rendimiento *(Min et al., 2022; Webson y Pavlick, 2022)*.

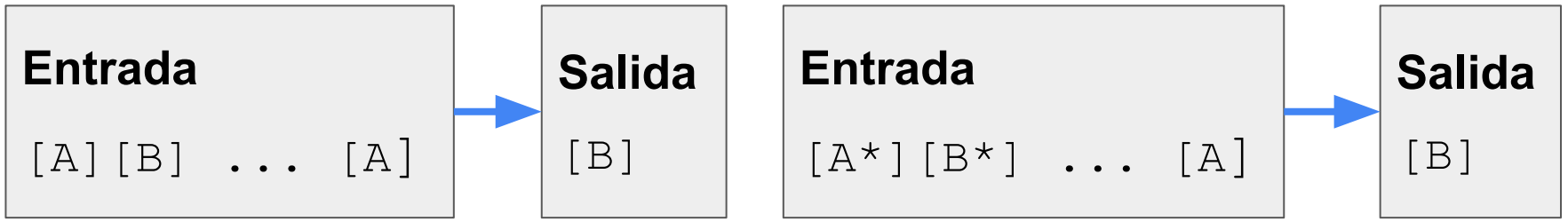
¿Cuáles ejemplos?

- Seleccionados de un conjunto etiquetado de entrenamiento.
- Usar demostraciones *similares* al ejemplo particular puede mejorar el rendimiento.
- Elegir ejemplos que maximicen el rendimiento sobre un conjunto de validación.

Cabezales de inducción (induction heads)

Cabezales de inducción: Pares de cabezales de atención de diferentes capas que trabajan en conjunto para copiar o completar patrones.





$A^* \approx A,\ B^* \approx B$

A Mathematical Framework for Transformer Circuits
Elhage et al., (2021)

In-context Learning and Induction Heads
Olsson et al., (2022)


Chain-of-Thought

Chain-of-Thought (CoT) Prompting

Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

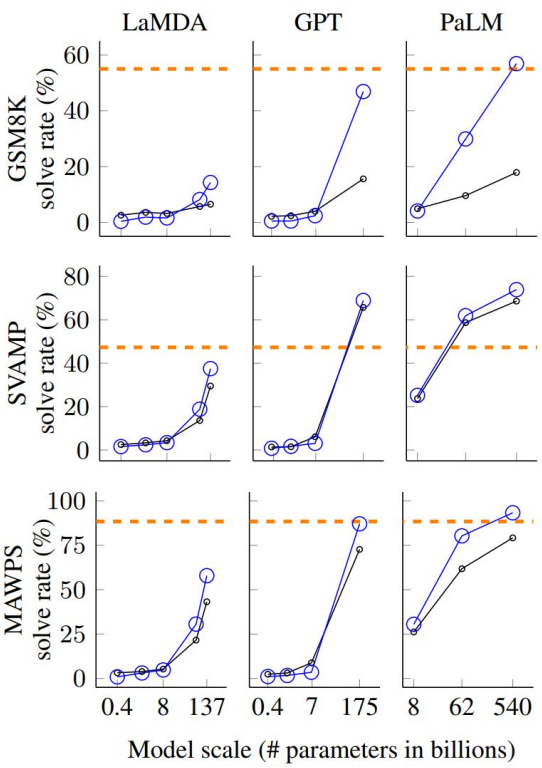
Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

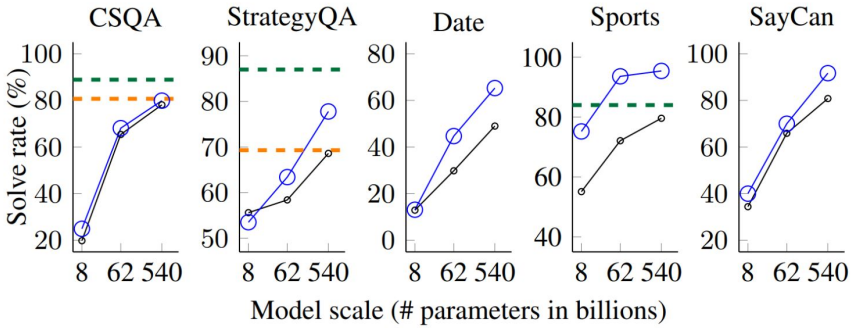
Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
Wei et al., (2022)

Chain-of-Thought (CoT) Prompting

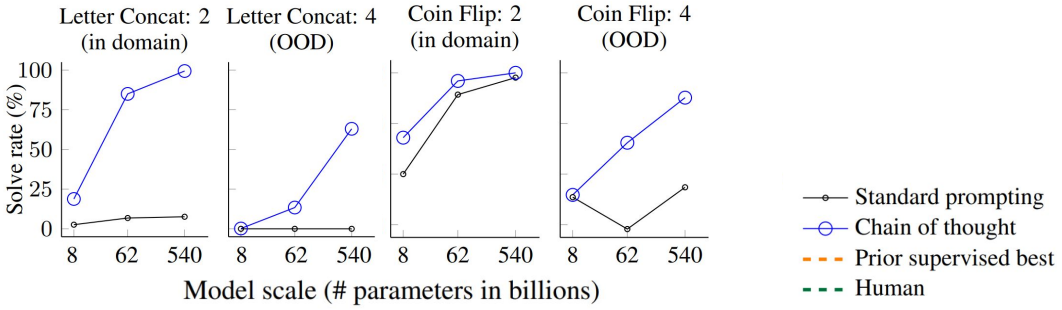
Razonamiento aritmético



Razonamiento de sentido común



Razonamiento simbólico



Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
Wei et al., (2022)

Self-consistency CoT

P: Si hay 3 autos en el estacionamiento y llegan 2 autos más, ¿cuántos autos hay en el estacionamiento?

R: Ya hay 3 autos en el estacionamiento. Llegan 2 más. Ahora hay 3 + 2 = 5 autos. La respuesta es 5.

…

P: Los gallinas ponen 16 huevos por día. Se comen tres en el desayuno y luego se hacen pasteles con cuatro. Vende el resto a $2 por huevo. ¿Cuánto gana cada día?

R:

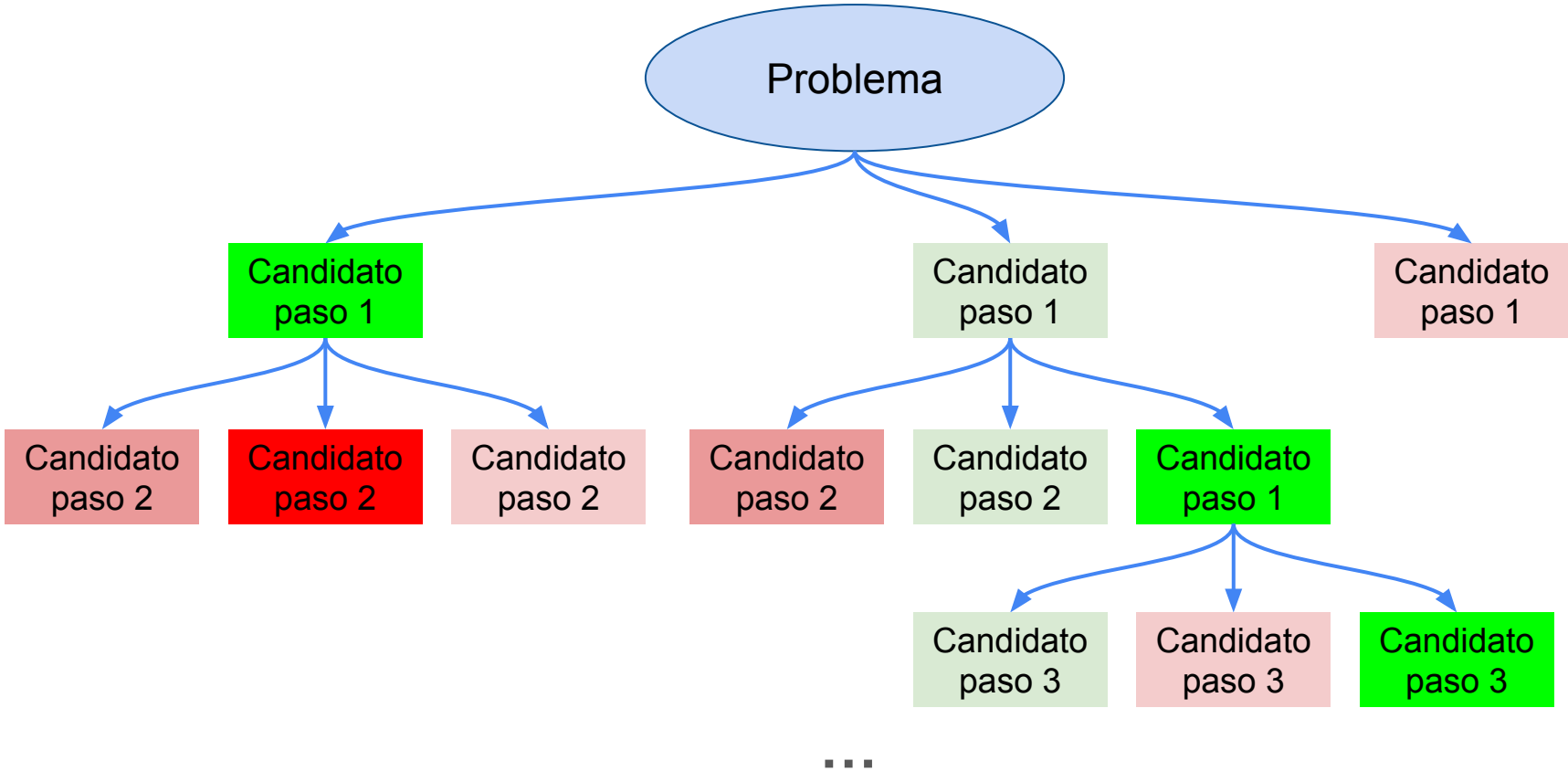
Le quedan 16 - 3 - 4 = 9 huevos. Por lo tanto, gana $2 * 9 = $18 por día. **La respuesta es $18.**

Esto significa que vende el resto por $2 * (16 - 4 - 3) = $26 por día. **La respuesta es $ 26.**

Come 3, por lo que le quedan 16 - 3 = 13. Luego hace pasteles, por lo que le quedan 13 - 4 = 9. Por lo tanto, le quedan 9 * $2 = $18. **La respuesta es 18$.**

Self-Consistency Improves Chain of Thought Reasoning in Language Models
Wang et al., (2023)

Tree-of-Thoughts (ToT) Prompting



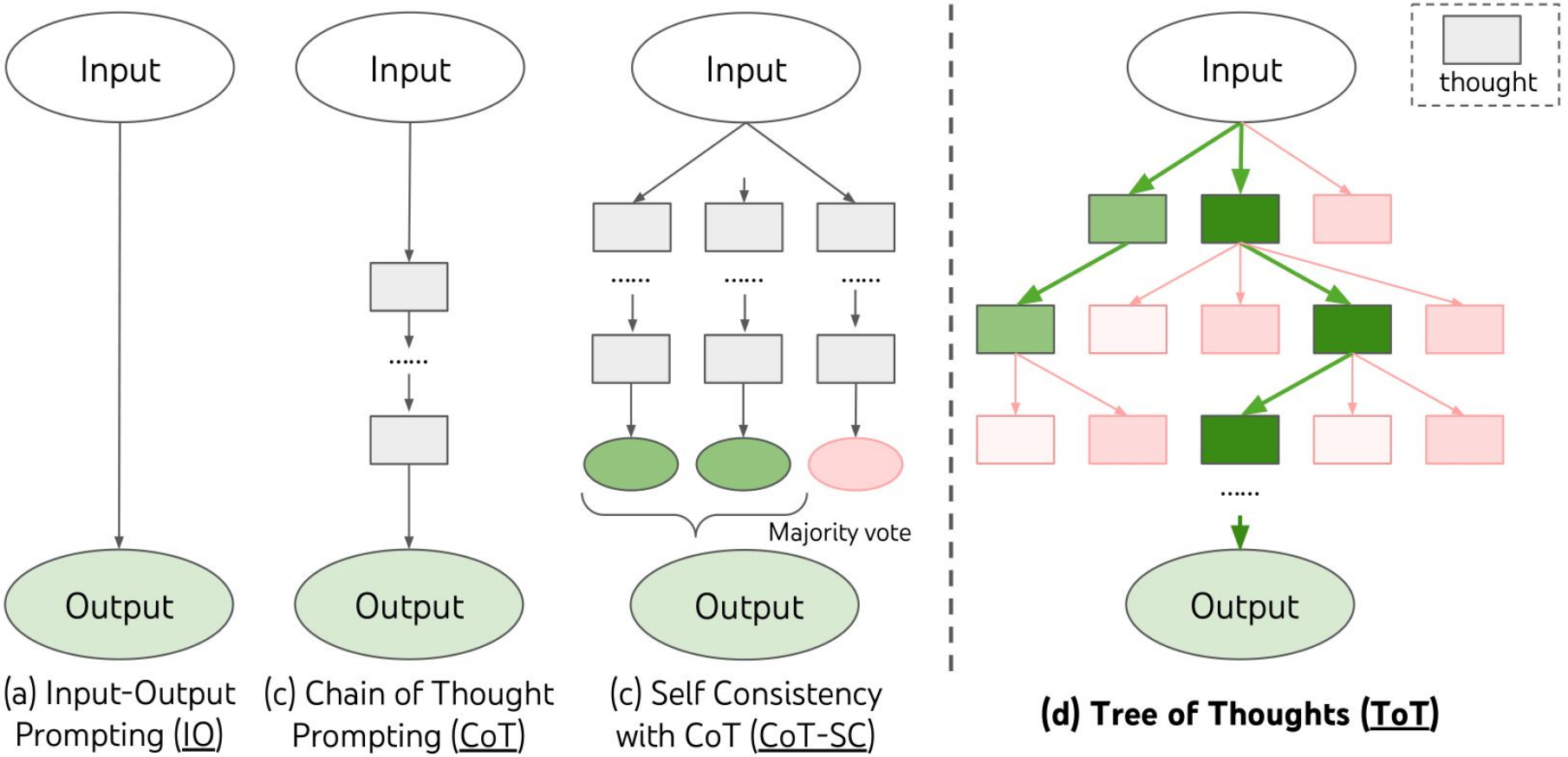
Tree of Thoughts: Deliberate Problem Solving with Large Language Models
Yao et al., (2023)

Tree-of-Thoughts (ToT) Prompting

1. **Descomposición en razonamientos:** Diseñar cómo son los pasos de razonamiento intermedios (por ej.: oración, ecuación, párrafo, etc).
2. **Generador de razonamientos:** Estrategia para generar los *k* candidatos para el siguiente razonamiento.
3. **Evaluador de estados:** Heurística para determinar qué razonamientos explorar y en qué orden. Usar también el modelo de lenguaje.
    a. **Etiquetar** cada razonamiento (por ej.: sure/likely/impossible).
    b. **Votar** sobre todos los razonamientos candidatos.
4. **Algoritmo de búsqueda:** Usar un algoritmo de búsqueda sobre los árboles (por ej.: BFS, DFS, etc).

Tree of Thoughts: Deliberate Problem Solving with Large Language Models
Yao et al., (2023)

CoT, CoT SC, ToT



Tree of Thoughts: Deliberate Problem Solving with Large Language Models
Yao et al., (2023)


Retrieval-Augmented Generation

Question Answering

Sistemas capaces de **responder preguntas** en lenguaje natural **de forma automática**.

- **QA extractivo:** extraer respuesta de un contexto. La respuesta está contenida en el contexto.
- **QA generativo:** generar la respuesta como texto libre.
  - **Abierto:** se provee información como contexto.
  - **Cerrado:** la respuesta se genera sin contexto.

LLMs para QA

¿Pueden los LLMs funcionar como sistemas para **QA generativo cerrado**?

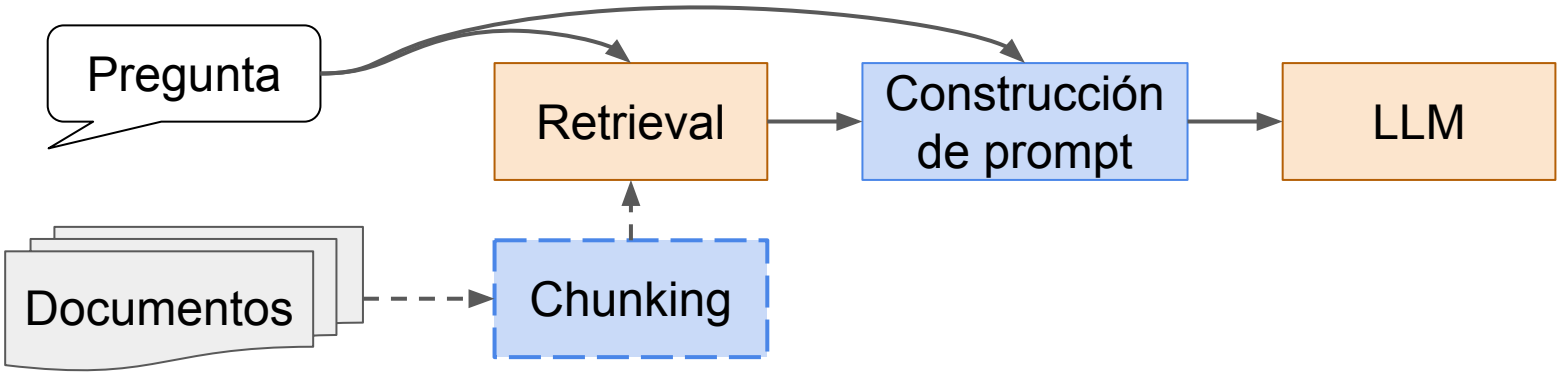
- Funcionan muy bien para responder sobre información muy presente en el conjunto de preentrenamiento.
- No sucede lo mismo con la información que se encuentra de forma mucho más esporádica (*long-tail knowledge*).
- No los podemos usar así para responder preguntas de un dominio específico…
- **Problemas:** alucinaciones, razonamientos erróneos, falta de referencias, información desactualizada, etc.

Large Language Models Struggle to Learn Long-Tail Knowledge
Kandpal et al., (2023)

Retrieval-Augmented Generation (RAG)

Método para **QA generativo abierto**, que se compone de dos etapas:

- **Recuperación de información:** recuperar la información más relevante a una consulta a partir de una base de documentos.
- **Generación de respuesta:** usar la información recuperada como contexto para que el LLM responda.



Modelos de IR con vectores densos

Chunking: Segmentar documentos en unidades más pequeñas llamadas *chunks*.

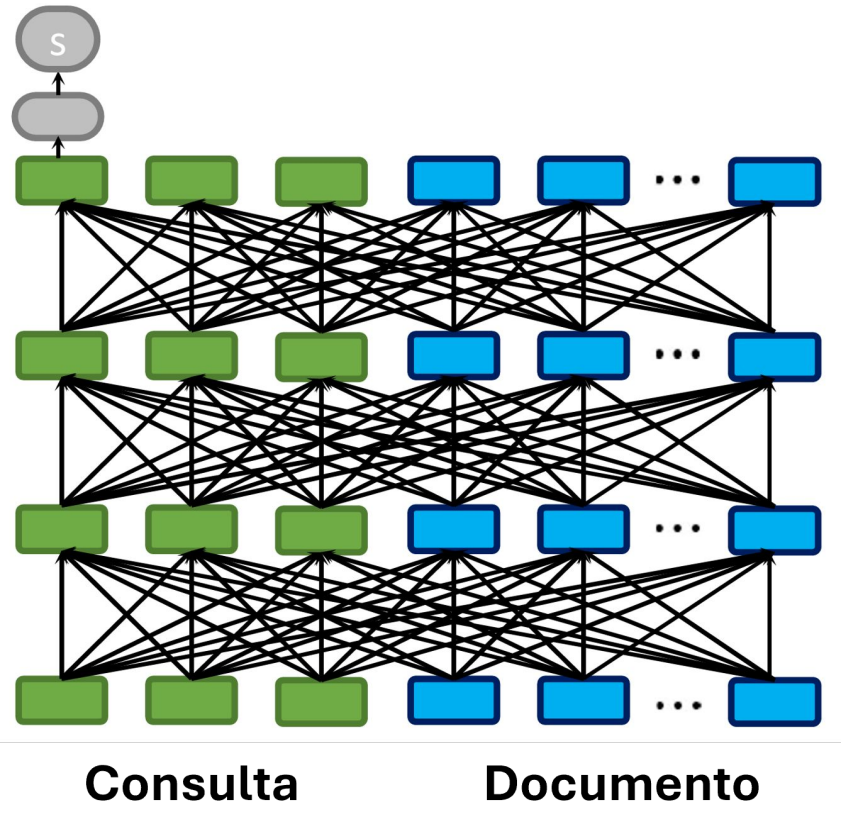
Queremos **recuperar los *chunks* más relevantes** a la pregunta.

Posible solución: Usar modelos de lenguaje enmascarados (por ej: BERT) adaptados (fine-tuning) para esta tarea:

- Cross-Encoder
- Bi-Encoder
- ColBERT
- Otros…

Cross-Encoder

- Un solo encoder **BERT**.
- Permite **interacción cruzada** entre consulta y documento.
- Dada una consulta, **se debe pasar cada *chunk*** junto a la consulta por el modelo.

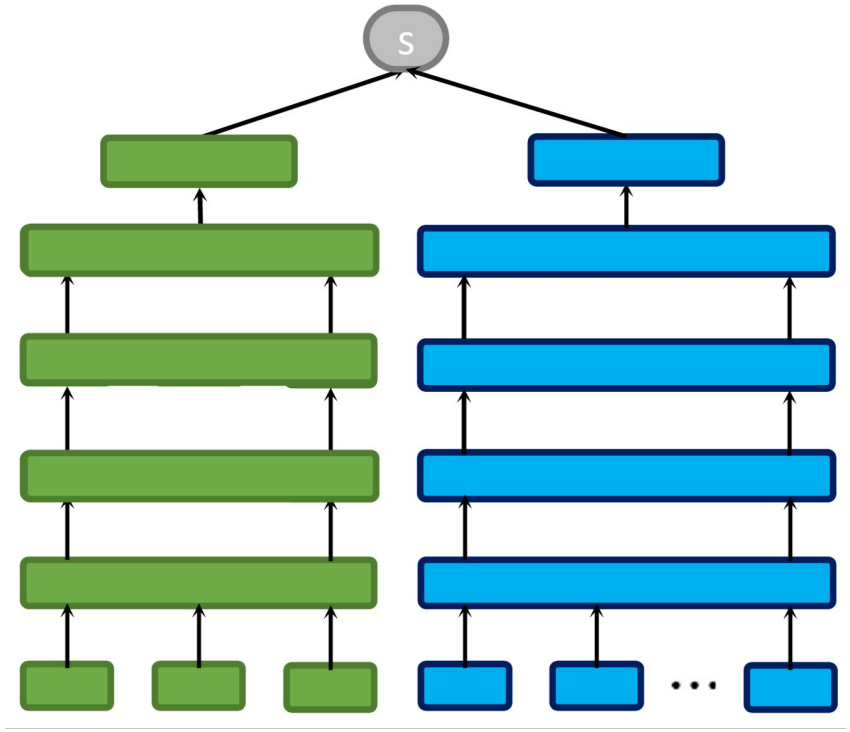


$$z = \text{BERT}(q; [\text{SEP}]; d)[\text{CLS}]$$

$$\text{score}(q, d) = \text{softmax}(U(z))$$

Bi-Encoder

- Dos encoders **BERT** (pueden ser el mismo).
- Se obtienen **representaciones** (*sentence embeddings*) de los textos.
- Se generan los **embeddings de los *chunks*** y se almacenan.
- Dada una consulta, se genera su embedding y **se calcula la distancia** con los *chunks*.

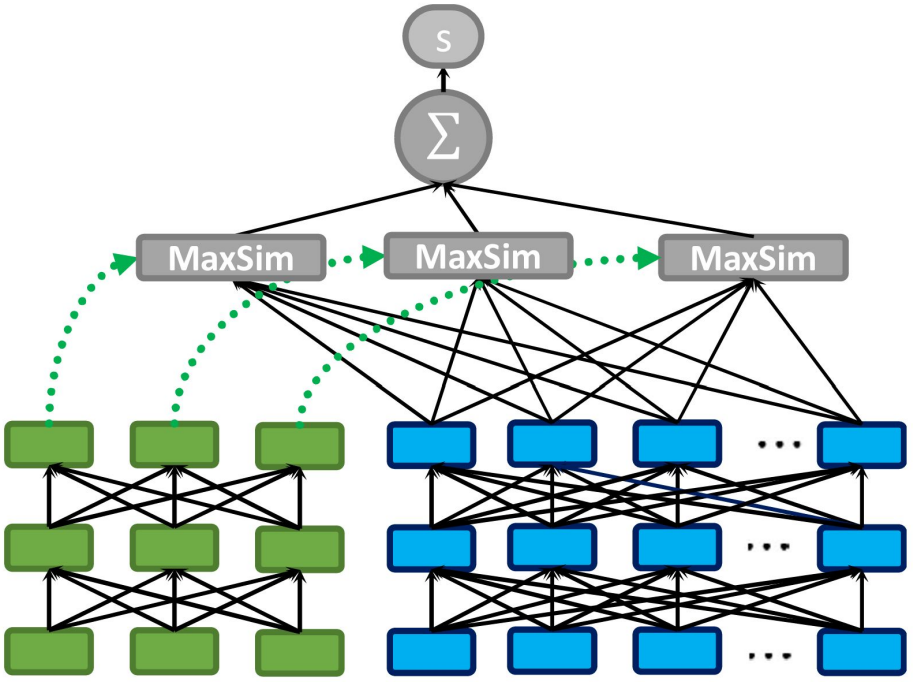


$$z_q = \text{BERT}_Q(q)[\text{CLS}]$$
$$z_d = \text{BERT}_D(d)[\text{CLS}]$$
$$\text{score}(q, d) = z_q \cdot z_d$$

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks
Reimers y Gurevych (2019)

ColBERT

- Dos encoders **BERT** (mismo modelo).
- Se obtienen **representaciones de cada token**, “interacción tardía”.
- Se generan los **embeddings de los tokens de los *chunks*** y se almacenan.
- Dada una consulta, se generan sus embeddings y **se calcula un score** para cada *chunk*.



$$\text{score}(q,d) = \sum_{i=1}^{N} \max_{j=1}^{m} \mathsf{E}_{q_i} \cdot \mathsf{E}_{d_j}$$

ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT
Khattab y Zaharia (2020)

Construcción de la prompt

Prompt

Indicaciones generales

Mantener una conversación sobre extractos de documentos relacionados con {*Tema*}. Dar respuestas cortas, concretas y precisas. Utilizar únicamente la información de los siguientes extractos para construir la respuesta:

***Chunks* recuperados**

Extracto de página web de título: {*Título del documento*} - URL: {*URL del documento*}
{*Texto del chunk*}

…

Extracto de página web de título: {*Título del documento*} - URL: {*URL del documento*}
{*Texto del chunk*}

Instrucciones específicas

Si los extractos no tienen una relación evidente con la pregunta, ignorarlos y responder que no se cuenta con información para responder la pregunta.

Indicaciones de formato

Siempre agregar al final de la respuesta los links a las referencias que se utilizaron en la respuesta bajo el título de “**Referencias:**”.

Bibliografía

- Jurafsky & Martin, 3rd Ed. (draft) - Capítulos 12 y 14
- Papers…