



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY



FACULTAD DE
INGENIERÍA

Búsqueda de respuestas en documentos institucionales utilizando grandes modelos de lenguaje

Informe de Proyecto de Grado presentado por

Ignacio Sastre

en cumplimiento parcial de los requerimientos para la graduación de la carrera
de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de
la República

Supervisores

Aiala Rosá
Guillermo Moncecchi

Montevideo, 24 de septiembre de 2024



Búsqueda de respuestas en documentos institucionales
utilizando grandes modelos de lenguaje por Ignacio Sastre
tiene licencia [CC Atribución 4.0](https://creativecommons.org/licenses/by/4.0/).

Agradecimientos

Agradezco en primer lugar a Aiala y Guillermo por su guía y apoyo. Gracias por siempre transmitirme confianza, incluso cuando me descolgaba con ideas locas. Fue muy importante sentir ese respaldo.

Agradezco también a Victoria García y Carlos Luna por su disposición y colaboración, como referentes del Espacio de Orientación y Consulta.

Gracias al Grupo PLN del Inco por acompañar durante todo el desarrollo del proyecto, y por brindar sus conocimientos y experiencia. También agradezco a todos los compañeros de trabajo, tanto de la Facultad como de Pyxis, por ayudarme a crecer y aprender.

Finalmente, agradezco a mi familia por ser siempre mis mayores maestros en la vida, y a mis amigos y compañeros por todo el apoyo durante estos años de carrera.

Resumen

Con la llegada de los grandes modelos de lenguaje (*Large Language Model - LLM*), los modelos conversacionales están convirtiéndose de manera progresiva en una alternativa interesante para interactuar con diferentes fuentes de información utilizando únicamente el lenguaje natural. Sin embargo, estos modelos enfrentan dificultades importantes, como las alucinaciones o razonamientos incorrectos, o la imposibilidad de presentar las fuentes de información utilizadas, lo que se hace más evidente en tareas acotadas a dominios específicos, como puede ser responder preguntas sobre un tema particular.

Este proyecto se centra en la exploración e implementación de métodos que permiten acotar la generación de respuestas de los LLMs en contextos específicos. La motivación principal es construir una herramienta que asista al Espacio de Orientación y Consulta (EOC) de la Facultad de Ingeniería de la Universidad de la República en la tarea de responder consultas de estudiantes, utilizando una base de documentos institucionales como fuente de información.

Se exploran varios métodos, como Retrieval-Augmented Generation (RAG), Fine-tuning de LLMs, y la utilización de Grandes ventanas de contexto. Se propone un nuevo método llamado Prompting de Derivaciones, basado en el concepto de derivaciones lógicas, que permite guiar la generación de respuestas de los LLMs. Se evalúan los métodos sobre un conjunto de preguntas y respuestas reales y se realiza un análisis de los resultados obtenidos.

Finalmente, se desarrolla un prototipo de una herramienta web que permite realizar consultas sobre la Facultad de Ingeniería utilizando los diferentes métodos explorados. Este prototipo está siendo evaluado por el EOC para brindar retroalimentación y validar su utilidad en un contexto real.

Palabras clave: Procesamiento de Lenguaje Natural, Búsqueda de respuestas, Recuperación de Información, Grandes Modelos de Lenguaje, Retrieval-Augmented Generation

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Objetivos	2
1.3. Organización del documento	2
2. Estado del arte	5
2.1. Arquitectura Transformer	7
2.1.1. Ventana de contexto	9
2.2. Fine-tuning	10
2.3. Retrieval Augmented Generation	12
2.3.1. Retrieval	13
2.3.2. Ventana de contexto y RAG	17
2.4. Técnicas de prompting	17
2.4.1. Lógica y Chain-of-Thought	21
2.5. Métricas de evaluación	22
2.5.1. LLMs como evaluadores (LLM-as-a-Judge)	24
2.6. Síntesis	24
3. Búsqueda de respuestas en documentos institucionales	27
3.1. Visión general de los métodos propuestos	30
3.2. Metodología de trabajo	32
3.3. Creación del corpus de trabajo	33
3.3.1. Selección y procesamiento de los documentos	33
3.3.2. Generación de <i>chunks</i>	34
3.4. Elección de modelos de lenguaje	36
3.5. Síntesis	38
4. Implementación de los métodos propuestos	41
4.1. Retrieval Augmented Generation	41
4.1.1. Retrieval	42
4.1.2. Generation	44
4.1.3. Cantidad de <i>chunks</i>	44
4.2. Fine-Tuning de LLM	45
4.2.1. Generación de conjunto de datos sintético	46

4.2.2.	Fine-tuning del modelo de lenguaje	51
4.3.	Prompting de derivaciones	52
4.3.1.	Definición del método	53
4.3.2.	Reglas	54
4.3.3.	Algoritmo	56
4.3.4.	LLMs utilizados	57
4.4.	Grandes ventanas de contexto	57
4.4.1.	Construcción de la prompt	58
4.4.2.	LLMs utilizados	59
4.4.3.	Tiempo de respuesta y costo	59
4.5.	Síntesis	60
5.	Evaluación y Resultados	61
5.1.	Evaluación	61
5.1.1.	Dataset de evaluación	61
5.1.2.	Métricas de evaluación	63
5.2.	Resultados	67
5.2.1.	Retrieval-Augmented Generation	69
5.2.2.	Fine-tuning	72
5.2.3.	Prompting de derivaciones	77
5.2.4.	Grandes ventanas de contexto	79
5.3.	Evaluación humana	81
5.4.	Síntesis	84
6.	Construcción del prototipo	87
6.1.	Decisiones a partir de la experimentación	87
6.2.	Arquitectura del prototipo	88
6.2.1.	Módulo QA	89
6.2.2.	Módulo Documents	90
6.3.	Interfaz de usuario	92
6.4.	Síntesis	93
7.	Conclusiones y Trabajo Futuro	95
7.1.	Conclusiones	95
7.2.	Trabajo Futuro	96
7.2.1.	Retrieval-Augmented Generation	96
7.2.2.	Fine-tuning de LLM	97
7.2.3.	Prompting de Derivaciones	97
7.2.4.	Grandes ventanas de contexto	98
7.2.5.	Metodología de evaluación	98
	Referencias	101

A. Prompts	109
A.1. Prompt utilizada en los ejemplos que no usan fuentes de información externas	109
A.2. Prompt utilizada en RAG	110
A.3. Prompts utilizadas en fine-tuning	110
A.3.1. Enfoque “Utilizar <i>chunks</i> necesarios”	110
A.3.2. Enfoque “Utilizar todos los <i>chunks</i> ”	111
A.3.3. Generación de preguntas fuera del dominio	112
A.4. Prompts utilizadas en prompting de derivaciones	113
A.4.1. Prompt de derivación completa	113
A.5. Prompt utilizada en grandes ventanas de contexto	116
A.6. Prompt utilizada para la limpieza de los mails	116
A.7. Prompt utilizada para la evaluación	117
B. Documento de Estado del Arte (Octubre 2023)	119
B.1. Introducción	119
B.2. Ventana de contexto	120
B.3. Fine-tuning	123
B.4. Retrieval Augmented Generation	125
B.4.1. Retriever	126
B.4.2. Generator	127
B.4.3. Ventana de contexto y retrieval augmentation	128
B.5. Modelos de memorización explícita	129
B.5.1. Construcción de la colección	129
B.5.2. Inferencia	129
B.5.3. Elección del parámetro λ	130
B.6. Técnicas de prompting	131
B.7. Métricas de evaluación	135

Capítulo 1

Introducción

Con la llegada de los grandes modelos de lenguaje (*Large Language Model* - LLM), los modelos conversacionales están convirtiéndose de manera progresiva en una alternativa interesante para interactuar con diferentes fuentes de información utilizando únicamente el lenguaje natural. Sin embargo, estos modelos enfrentan dificultades importantes, como las alucinaciones o razonamientos incorrectos (Valmeekam, Olmo, Sreedharan, y Kambhampati, 2022; Ji y cols., 2023), o la imposibilidad de presentar las fuentes de información utilizadas, lo que se hace más evidente en tareas acotadas a dominios específicos, como puede ser responder preguntas sobre un tema particular.

Para mitigar estas dificultades, se han desarrollado diferentes técnicas que buscan acotar la generación de respuestas de los modelos conversacionales en pos de responder consultas tomando como fuente de información una base de documentos de un dominio específico (Gao y cols., 2024; P. Lewis, Perez, y cols., 2020; Guu y cols., 2020; Khandelwal y cols., 2019) (*inter alia*).

Por otro lado, la Facultad de Ingeniería de la Universidad de la República cuenta con el Espacio de Orientación y Consulta¹ (EOC), quienes se encargan, entre otras cosas, de responder las consultas de los estudiantes relacionadas con su proceso académico. Actualmente, los integrantes de este equipo responden las consultas recibidas manualmente. Técnicas como las mencionadas anteriormente pueden resultar oportunas para asistir en esta tarea, facilitando el trabajo y agilizando los tiempos de respuesta.

1.1. Contexto

Este proyecto se realiza con la colaboración del Espacio de Orientación y Consulta, que se definen como un lugar de referencia durante la vida académica-estudiantil, y se encargan de atender consultas de forma personalizada, brindar información sobre la Facultad y sobre becas, así como asesorar sobre estrategias de aprendizaje y técnicas de estudio.

¹<https://www.fing.edu.uy/es/ensenanza/espacio-de-orientacin-y-consulta>

En particular, el EOC deja disponible un correo electrónico a través del cual se pueden realizar consultas para recibir la asistencia de un integrante del equipo.

Bajo este contexto, en este proyecto de grado se busca desarrollar una herramienta que asista en la generación de respuestas a consultas de estudiantes, ya sea para ponerla a disposición directamente a los estudiantes o para asistir a los funcionarios del EOC en la tarea de responder las consultas recibidas por correo.

1.2. Objetivos

Se plantean los siguientes objetivos principales para este proyecto:

- Exploración y desarrollo de múltiples técnicas que permitan acotar la información utilizada para la generación de respuestas de los modelos de lenguaje a un conjunto de documentos predefinido. Se desprenden entonces los siguientes objetivos específicos:
 - Estudio e implementación de los métodos existentes en la literatura;
 - Definición e implementación de posibles nuevos métodos para la resolución de este problema.
- Definición de un método de evaluación de los experimentos realizados, que permita analizar los resultados obtenidos y sacar conclusiones sobre los métodos explorados.
- Construcción de un prototipo de una herramienta que permita realizar consultas sobre la Facultad de Ingeniería para poner a disposición para el EOC, utilizando el o los métodos explorados que den mejores resultados en la etapa de evaluación.

La tecnología y los métodos que serán explorados en este proyecto son bastante recientes. La arquitectura *Transformer* (Vaswani y cols., 2017) que utilizan los LLMs se presentó en 2017, y el modelo GPT-3 (Brown y cols., 2020), que dio un salto cualitativo importante respecto a los modelos anteriores, se lanzó en 2020. Por lo tanto, es aún una pregunta abierta si estos modelos pueden ser utilizados de forma confiable y segura para el tipo de tarea que se plantea en este proyecto, en donde es fundamental no generar respuestas incorrectas que puedan derivar en desinformación que confunda a estudiantes. Es por esto que en este proyecto también se busca ofrecer de forma empírica argumentos que permitan acercarse a una respuesta a esta pregunta.

1.3. Organización del documento

El resto del documento se organiza de la siguiente manera:

- [Capítulo 2: Estado del Arte](#) presenta el estudio de la literatura llevado a cabo, e introduce los fundamentos teóricos de los métodos que se exploran a lo largo del proyecto.
- [Capítulo 3: Búsqueda de respuestas en documentos institucionales](#) presenta una visión general de los métodos propuestos, así como una descripción de la metodología de trabajo, los datos utilizados y los LLMs explorados.
- [Capítulo 4: Implementación de los métodos propuestos](#) detalla los experimentos realizados para cada uno de los métodos explorados.
- [Capítulo 5: Evaluación y Resultados](#) explica la metodología de evaluación llevada adelante, y presenta los resultados de los experimentos realizados en el capítulo anterior, así como un análisis de estos resultados para cada uno de los métodos explorados.
- [Capítulo 6: Construcción del prototipo](#) presenta las decisiones de análisis, diseño y construcción del prototipo desarrollado.
- [Capítulo 7: Conclusiones y Trabajo Futuro](#) expone las conclusiones obtenidas, así como las diferentes líneas que quedan abiertas para desarrollar o profundizar.

Capítulo 2

Estado del arte

En este capítulo se condensa el estado del arte de la utilización de grandes modelos de lenguaje causales para la tarea de búsqueda de respuestas (*question answering*), en el caso en que contamos con un conjunto de documentos D pertenecientes a un dominio específico sobre el cual se quiere responder preguntas. Este capítulo es un resumen y actualización del documento de Estado del Arte realizado en Octubre de 2023 y disponible en el Anexo B. Antes de continuar, se definirán los conceptos previos necesarios para entender el resto del capítulo.

Este trabajo se enmarca en el área de **Procesamiento de Lenguaje Natural (PLN)**, un campo interdisciplinario cuyo objetivo es lograr que las computadoras realicen tareas útiles que involucren lenguaje humano, como permitir la comunicación entre humanos y máquinas, mejorar la comunicación entre humanos, o realizar procesamientos útiles de texto o habla (Jurafsky y Martin, 2024).

Entre los diversos enfoques posibles para el problema abordado, en este proyecto trabajamos principalmente con modelos de lenguaje. Un **modelo de lenguaje** es un modelo que asigna probabilidades a secuencias de palabras (Jurafsky y Martin, 2024). Se representará una secuencia w de n palabras como w_1, w_2, \dots, w_n , o de forma reducida como $w_{1:n}$. Para la probabilidad conjunta de cada palabra en una secuencia teniendo un valor particular $P(X_1 = w_1, X_2 = w_2, \dots, X_n = w_n)$ se utilizará la notación $P(w_1, w_2, \dots, w_n)$.

Utilizando la regla de la cadena de la probabilidad, se puede descomponer la probabilidad de la secuencia w de la siguiente manera:

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_n|w_{1:n-1})$$

Esta descomposición muestra que la tarea de modelado de lenguaje se puede resolver sabiendo calcular la probabilidad condicionada de una palabra dadas las palabras anteriores en la secuencia $P(w_n|w_{1:n-1})$.

Llamamos **grandes modelos de lenguaje** (*Large Language Models* - LLM) a modelos de aprendizaje profundo que utilizan la arquitectura Transformer (Vaswani y cols., 2017) y cuentan con cientos o miles de millones de parámetros.

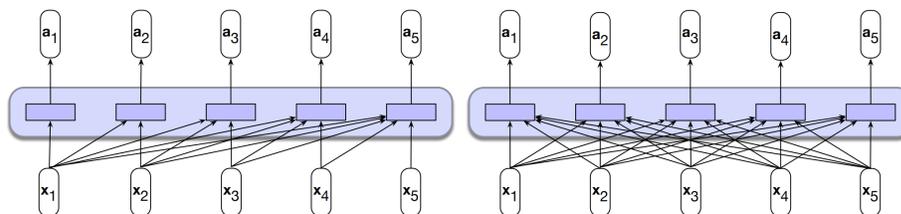


Figura 2.1: Izquierda: Capa de *self-attention* causal. Derecha: Capa de *self-attention* bidireccional. Figura extraída de [Jurafsky y Martin \(2024\)](#).

Algunos ejemplos conocidos son BERT ([Devlin, Chang, Lee, y Toutanova, 2019](#)) Llama ([Touvron, Lavril, y cols., 2023](#)), GPT-3 ([Brown y cols., 2020](#)), entre otros.

Existen dos tipos diferentes de grandes modelos de lenguaje: modelo de lenguaje causal y modelo de lenguaje enmascarado (bidireccional) (Figura 2.1).

Un **modelo de lenguaje causal** es entrenado para la tarea de generar tokens (unidad mínima de texto, suelen ser palabras o subpalabras) de forma incremental mediante la acción repetida de predecir el siguiente token condicionado por los tokens anteriores, que pueden ser parte del contexto o tokens previamente predichos por el propio modelo ([Jurafsky y Martin, 2024](#)). GPT-3 y Llama son ejemplos de modelos de este tipo.

Por otro lado, un **modelo de lenguaje enmascarado** se entrena de forma bidireccional. Esto es, tomando como contexto tanto los tokens anteriores como posteriores de la secuencia. Luego, se entrena el modelo de lenguaje para predecir el token más probable que sustituya a un token particular (máscara) presente en la secuencia de tokens.

La tarea de **Question Answering (QA)** refiere a la construcción de sistemas capaces de responder preguntas en lenguaje natural de forma automática. La mayoría de sistemas de QA se centran en responder preguntas factoides, esto es, preguntas que pueden ser respondidas con hechos simples expresados en textos cortos ([Jurafsky y Martin, 2024](#)).

Existen dos variantes de sistemas de QA ([HuggingFace, 2021](#)):

- **Extractive Question Answering:** El modelo debe extraer la respuesta de un contexto (por ejemplo, un conjunto de documentos). Esta variante se suele resolver utilizando modelos de lenguaje enmascarados, como BERT.
- **Generative Question Answering:** El modelo debe generar la respuesta como texto libre. Más precisamente, es posible clasificar estos sistemas en *Open Generative QA* si el modelo debe generar su respuesta a partir de un contexto, y *Closed Generative QA* si no se provee un contexto y la respuesta es completamente generada por el modelo. Para esta tarea, los modelos de lenguaje causales son más indicados por su naturaleza generativa.

A continuación, en la Sección 2.1 se introducirá la arquitectura Transformer y se explorarán los trabajos relativos a modificar la arquitectura para procesar segmentos de texto más grandes. En la Sección 2.2 se comentará sobre técnicas de fine-tuning. En la Sección 2.3 se estudiarán las técnicas que utilizan recuperación de información para seleccionar los documentos adecuados y modelos de lenguaje causales para generar la respuesta a partir de estos. En la Sección 2.4 se explorarán las técnicas de prompting relevantes a la tarea de *Generative Question Answering*. Por último, en la Sección 2.5 se realizará una exploración sobre los métodos de evaluación para esta tarea.

2.1. Arquitectura Transformer

La arquitectura Transformer (Vaswani y cols., 2017) es una arquitectura de aprendizaje profundo que se ha convertido en la base de los grandes modelos de lenguaje actuales. El Transformer, como se propuso en (Vaswani y cols., 2017), es una arquitectura Encoder-Decoder que utiliza el mecanismo atencional (Bahdanau, Cho, y Bengio, 2015) como componente principal, en lugar de las capas recurrentes o convolucionales utilizadas en arquitecturas previas. La Figura 2.2 muestra dicha arquitectura.

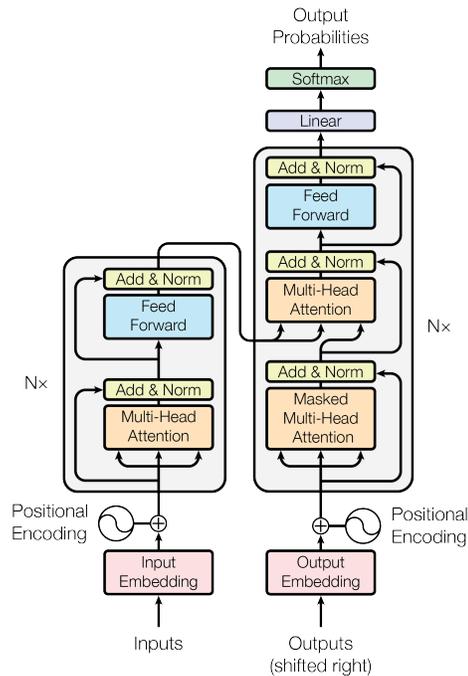


Figura 2.2: Arquitectura Transformer. Figura extraída de Vaswani y cols. (2017).

El Transformer se compone de una pila de bloques de Transformer (*Transfor-*

mer blocks). Cada uno de ellos recibe como entrada una secuencia de vectores x_1, x_2, \dots, x_n y la transforma en una nueva secuencia z_1, z_2, \dots, z_n del mismo largo. Cada bloque de Transformer combina capas de *self-attention* y capas *feed-forward* (capa densa completamente conectada), junto a normalizaciones y conexiones residuales (pasan la entrada de una capa a la siguiente sin modificaciones). Además, una capa inicial de *word embeddings* se utiliza para obtener representaciones vectoriales densas de cada token de la secuencia de entrada, los cuales se utilizan como entrada al primer bloque de Transformer (Jurafsky y Martin, 2024).

La capa de *self-attention* permite generar una representación contextualizada de cada token en la secuencia, tomando en cuenta la información de todos los tokens de la secuencia. Cada embedding de entrada x_i es proyectado en tres vectores: consulta (q_i - *query*, con una matriz de pesos Q), clave (k_i - *key*, con una matriz de pesos K) y valor (v_i - *value*, con una matriz de pesos V). Luego, para cada token x_i , se calcula una combinación lineal de los valores v_j ponderados por la similitud entre la consulta q_i con cada clave k_j . La similitud se calcula como el producto escalar entre q_i y k_j (escalado por un factor relativo a la dimensión de los vectores) y se aplica la función *softmax* para obtener los pesos de la combinación lineal α_{ij} , a los que llamamos pesos de atención.

Intuitivamente, el peso de atención α_{ij} indica cuán relevante es el token j para el token i . Por ejemplo, en la frase “El perro no quiere comer, él ya está lleno”, el peso de atención del token “él” respecto al token “perro” debería ser alto, ya que el pronombre “él” se refiere al sustantivo “perro”. La Figura 2.3 muestra de forma gráfica los pesos de atención para el ejemplo anterior, de uno de los bloques de *self-attention* de la primer capa del modelo de lenguaje Beto (Cañete y cols., 2020).

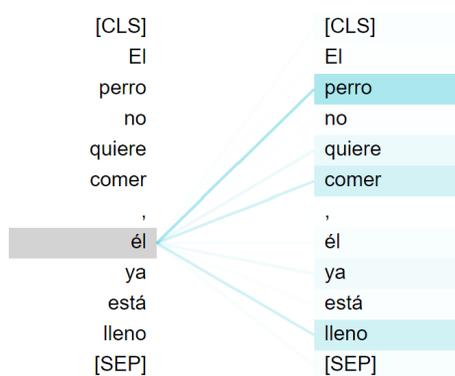


Figura 2.3: Ejemplo gráfico de los pesos de atención para un bloque de *self-attention* de la primer capa del modelo de lenguaje Beto, generada con la herramienta BertViz (Vig, 2019).

Multi-Head Attention refiere a conjuntos de capas de *self-attention* (llamadas *heads*) que se calculan en paralelo en una misma capa, cada uno con sus propias

proyecciones de consulta, clave y valor. Las salidas de cada *head* se concatenan y se proyectan linealmente con una capa *feed-forward* para obtener la salida final.

Los *word embeddings* que se utilizan al inicio de la arquitectura Transformer no capturan la información de la posición de los tokens en la secuencia. Para solucionar esto, se definen los vectores de posición (*positional embeddings*) que se suman a los *word embeddings* antes de ser utilizados como entrada al primer bloque de Transformer.

Hay diferentes maneras de definir estos vectores de posición. El método más simple llamado posición absoluta (*absolute position*) consiste en definir un vector de posición distinto para cada posición en la secuencia (hasta un largo fijo), y aprender estos vectores de posición durante el entrenamiento (Jurafsky y Martín, 2024). Otra alternativa, utilizada en (Vaswani y cols., 2017), es definir los vectores de posición como combinaciones de senos y cosenos de diferentes frecuencias, capturando las relaciones inherentes entre las posiciones.

2.1.1. Ventana de contexto

Los modelos de lenguaje basados en la arquitectura Transformer tienen una ventana de contexto (largo máximo de la entrada) con un tamaño fijo y acotado. Por ejemplo, Llama 2 (Touvron, Martin, y cols., 2023) tiene una ventana de contexto de 4096 tokens, lo que equivale a aproximadamente 3000 palabras en inglés.

Esta limitación se debe a que la implementación estándar del modelo atencional tiene complejidad computacional $O(n^2d)$ y de memoria $O(n^2 + nd)$, donde n es el largo de la ventana de contexto y d es la dimensión de representación interna (Vaswani y cols., 2017). Esto limita el tamaño de la ventana de contexto a elegir al diseñar un modelo Transformer.

En el problema de *Open Generative QA*, tener una ventana de contexto más grande posibilitaría, por ejemplo, utilizar de manera directa el modelo de lenguaje causal para generar la respuesta a una entrada (pregunta) x , pasando al modelo la concatenación de todos los documentos D y la entrada x como contexto: $P(y|D, x)$. Con una ventana de contexto relativamente pequeña, como la de Llama 2, esto no sería posible de implementar si se cuenta con varios documentos o documentos extensos. Múltiples trabajos buscan solventar esta limitación.

Transformer-XL (Dai y cols., 2019) es una arquitectura que introduce la noción de recurrencia a la arquitectura Transformer. Se reutilizan los estados ocultos de los segmentos previos a la hora de computar el segmento actual, construyéndose una conexión recurrente entre segmentos. Para lograr esto, es también necesario calcular de forma diferente los *positional embeddings*, pasando de una representación absoluta a una relativa.

Otros trabajos se centran en modificar el modelo atencional para mejorar su eficiencia computacional y de memoria. En la arquitectura *Reformer* (Kitaev, Kaiser, y Levskaya, 2019) se computa la atención de manera aproximada mediante el uso de *locality-sensitive hashing (LSH)*, que agrupa elementos similares en los mismos *buckets* con alta probabilidad. Así, solo se presta atención a

elementos del mismo *bucket*. Esto permite cambiar el factor $O(n^2)$ de la complejidad de la atención por $O(n \log n)$.

Un enfoque distinto se presenta en [Dao, Fu, Ermon, Rudra, y Re \(2022\)](#), bajo el argumento de que los algoritmos propuestos hasta el momento no tienen en cuenta las lecturas y escrituras (*IO*) que se realizan en la memoria. La mayoría de las operaciones de los Transformers están limitadas por los accesos a la memoria y no por el cómputo.

Con esta motivación se propone *FlashAttention* ([Dao y cols., 2022](#)), un algoritmo que computa la atención de manera exacta con menos accesos a memoria. [Dao \(2023\)](#) propone luego *FlashAttention-2* que, bajo la misma motivación que el anterior, logra mejorar aproximadamente el doble en la velocidad de cómputo respecto a *FlashAttention*, acercándose al máximo rendimiento (*throughput*) alcanzable por la GPU. El uso de *FlashAttention-2* permitió el desarrollo de *togethercomputer/LLaMA-2-7B-32K*¹, un modelo que incrementa a 32 mil tokens la ventana de contexto de Llama 2 (originalmente de 4096 tokens).

2.2. Fine-tuning

Fine-tuning es el proceso de tomar una red neuronal pre-entrenada (en este caso un LLM) y continuar el entrenamiento, muchas veces agregando una capa de clasificación que toma la capa superior del modelo como entrada, para realizar una tarea específica (*downstream task*) como puede ser *Named Entity Recognition* o *Question Answering*. La intuición es que la etapa de pre-entrenamiento sirve para aprender representaciones complejas del lenguaje, y esto facilita que el modelo sea adaptado (fine-tuning) a los requerimientos de una tarea específica ([Jurafsky y Martin, 2024](#)).

Realizar fine-tuning puro sobre un LLM trae algunos inconvenientes que pueden ser difíciles de sobrellevar. En primer lugar, el resultado de esta tarea es un nuevo modelo con coeficientes diferentes al modelo original y por ende es necesario almacenarlo. Debido a la enorme cantidad de parámetros (la familia **Llama 2** tiene modelos de entre 7 mil millones y 70 mil millones de parámetros), resulta inviable mantener nuevos modelos de esta envergadura cada vez que se realiza un fine-tuning. Por otro lado, realizar el fine-tuning es muy costoso computacionalmente y requiere mucho uso de memoria debido a la necesidad de ajustar todos los coeficientes del modelo pre-entrenado. Esta limitación hace que sea prohibitivo realizar fine-tuning sin usar hardware específico y costoso.

Para mitigar estas dificultades, se han propuesto distintos métodos que buscan mejorar la eficiencia del proceso de fine-tuning así como disminuir la cantidad de los parámetros a ajustar y almacenar.

Una técnica propuesta es *Low Rank Adaptation (LoRA)* ([Hu y cols., 2021](#)). Para explicar este método, se recuerda que el rango de una matriz es la cantidad de filas linealmente independientes. Luego, la descomposición por rango de una matriz $A \in \mathbb{M}_{m \times n}$ de rango r garantiza que existen matrices $C \in \mathbb{M}_{m \times r}$ y $F \in \mathbb{M}_{r \times n}$ tales que $A = CF$.

¹<https://huggingface.co/togethercomputer/LLaMA-2-7B-32K>



Figura 2.4: Funcionamiento del modelo actualizado (fine-tuning) en etapa de inferencia utilizando *Low-Rank Adapters*. Figura traducida y adaptada de [Raschka \(2023\)](#).

El método *LoRA* consiste en congelar los coeficientes del modelo preentrenado e inyectar matrices de descomposición por rango entrenables en cada capa de la arquitectura Transformer. Esto permite reducir la cantidad de coeficientes entrenables, permitiendo disminuir los requerimientos de cómputo y de memoria así como de espacio de almacenamiento, ya que alcanza con almacenar solamente las matrices de descomposición de rango, que poseen muchos menos parámetros que el modelo original.

Por ejemplo, supongamos que $W \in \mathbb{M}_{m \times n}$ es una matriz de pesos y $\Delta W \in \mathbb{M}_{m \times n}$ es la matriz de actualizaciones que queremos realizar a los pesos. En otras palabras, lo que se busca al realizar fine-tuning, en este ejemplo de una sola matriz de pesos, es aprender una matriz de pesos $W' \in \mathbb{M}_{m \times n}$ que resulta de modificar a la matriz de pesos pre-entrenada W . Esa diferencia resultante entre W y W' es representada por ΔW , por lo que se cumple que $W' = W + \Delta W$.

Como se muestra en [Raschka \(2023\)](#), en lugar de aprender ΔW , en *LoRA* se propone descomponer esta matriz en dos matrices de menor tamaño: $\Delta W = W_m W_n$, donde $W_m \in \mathbb{M}_{m \times r}$, $W_n \in \mathbb{M}_{r \times n}$, y r es un rango pequeño impuesto para la matriz ΔW . Manteniendo los pesos originales congelados y solamente entrenando estas nuevas matrices permite reducir la cantidad de coeficientes entrenables de $m * n$ a $m * r + r * n$. Si r es un valor pequeño, la diferencia puede ser muy considerable. Luego del entrenamiento, podemos reconstruir la matriz de pesos del modelo con fine-tuning W' sumando a los pesos originales W el producto de las dos matrices aprendidas: $W' = W + W_m W_n$. La figura 2.4 muestra el funcionamiento del modelo con fine-tuning en etapa de inferencia.

En *QLoRA* ([Dettmers, Pagnoni, Holtzman, y Zettlemoyer, 2023](#)) se toma esta idea y se aplica sobre modelos cuantizados. Cuantizar un modelo significa representar los coeficientes y activaciones del modelo utilizando tipos de datos de poca precisión como enteros de 8 bits (int8) o de 4 bits (int4) en lugar de usar

punto flotante de 32 bits (float32)². Esta reducción permite requerir una menor cantidad de memoria a la hora de utilizar el modelo en etapa de inferencia.

Sin embargo, no era posible realizar fine-tuning sobre los modelos ya cuantizados sin degradar el rendimiento. *QLoRA* es una técnica de cuantización a 4 bits que permite agregar un conjunto de matrices *Low Rank Adapter (LoRA)* que son ajustadas mediante backpropagation a través de los pesos ya cuantizados, sin degradación de rendimiento.

Esto reduce los requerimientos de memoria para realizar fine-tuning notoriamente. Por ejemplo, realizar fine-tuning sobre un modelo de 65 mil millones de parámetros requeriría más de 780GB de memoria de GPU. Mediante la técnica *QLoRA*, se reduce a menos de 48GB de memoria.

Independientemente del método de fine-tuning que se utilice, un cuestionamiento razonable a hacerse es si efectivamente se está mejorando el rendimiento del LLM en la tarea objetivo, en comparación con utilizar técnicas de prompting como *few-shot* (véase sección 2.4). En [Wei y cols. \(2022\)](#) se propone realizar fine-tuning con datasets adaptados de una manera particular, a lo que se le llama *instruction tuning*. Se toma más de 60 datasets de tareas de PLN y se adaptan para ser expresados mediante instrucciones en lenguaje natural.

La idea es que al enseñar al modelo a realizar tareas definidas mediante instrucciones, este será capaz de aprender a seguir instrucciones incluso para tareas no vistas en la etapa de fine-tuning. Para evaluar este comportamiento, se agrupan los datasets en *clusters* del mismo tipo de tarea, y se evalúa con cada uno de estos realizando *instruction tuning* con el resto de los *clusters*.

Mediante *instruction tuning*, en [Wei y cols. \(2022\)](#) se entrena el modelo FLAN a partir del modelo pre-entrenado LAMBDA-PT. El trabajo concluye que FLAN supera a LAMBDA-PT en todas las tareas e incluso supera a GPT-3 ([Brown y cols., 2020](#)) con *few-shot* en 10 datasets, demostrando la efectividad de realizar *instruction tuning*.

2.3. Retrieval Augmented Generation

Si bien se ha observado que los modelos de lenguaje preentrenados adquieren bastante información y conocimiento a partir de los datos de entrenamiento sin acceder a información externa ([Petroni y cols., 2019](#)), existen múltiples desventajas en el uso de estos modelos. Por ejemplo, no es posible actualizar el conocimiento en tiempo real ni obtener las fuentes a partir de las cuales se obtiene la información. Además, pueden generar alucinaciones (información que no se desprende de los datos de entrenamiento, inventada) ([Ji y cols., 2023](#)). La Figura 2.5 muestra un ejemplo de alucinación en donde el modelo GPT-4 inventa que el ex-presidente José Mujica ganó el Premio Nobel de la Paz en 2010.

Para mitigar estos problemas, una alternativa es usar el método de *Retrieval Augmented Generation (RAG)*, que consiste en utilizar un componente de recuperación de información para recuperar la información más relevante a una

²https://huggingface.co/docs/optimum/concept_guides/quantization

Pregunta

¿Qué persona de nacionalidad uruguaya ganó un premio Nobel?

Respuesta (gpt-4-1106-preview)

La persona de nacionalidad uruguaya que recibió un Premio Nobel es José Mujica, quien fue galardonado con el Premio Nobel de la Paz en 2010. Mujica fue reconocido por su enfoque progresista en la presidencia de Uruguay y sus esfuerzos por promover la paz y la justicia social. Además, su estilo de vida austero y su enfoque en la igualdad y la inclusión social lo hicieron muy popular tanto en Uruguay como internacionalmente.

Figura 2.5: Ejemplo de alucinación del modelo de lenguaje GPT-4 (versión gpt-4-1106-preview).

consulta desde una base de documentos externa (P. Lewis, Perez, y cols., 2020; Izacard y cols., 2024; Shi y cols., 2024).

El paradigma RAG consiste, en su versión más sencilla, de dos etapas principales: Recuperación (*Retrieval*) y Generación (*Generation*). En primer lugar, los documentos son segmentados en unidades más pequeñas, a las que llamamos *chunks*, que son luego convertidas a representaciones vectoriales. Dada una consulta de usuario, durante la etapa de Retrieval la consulta se convierte a su representación vectorial y se calcula la distancia entre la consulta y los *chunks*. Los k *chunks* más cercanos son recuperados y usados como contexto en la etapa de generación (Gao y cols., 2024).

En P. Lewis, Perez, y cols. (2020) se formalizan estos dos componentes de la siguiente manera:

- **Retriever** $p_{\eta}(z|x)$, con parámetros η utilizado para obtener los k documentos z que están más relacionados con la entrada x .
- **Generator** $p_{\theta}(y_i|x, z, y_{1:i-1})$, con parámetros θ que cumple la tarea de modelo de lenguaje causal, basándose en los previos tokens generados $y_{1:i-1}$, la entrada x y el documento recuperado z .

La Figura 2.6 muestra los dos componentes mencionados.

Además, en P. Lewis, Perez, y cols. (2020), estos dos componentes luego son combinados y entrenados end-to-end mediante fine-tuning. Sin embargo, también es posible combinar ambos modelos utilizando técnicas de prompting sin necesidad de pasar por una etapa de entrenamiento.

2.3.1. Retrieval

En P. Lewis, Perez, y cols. (2020) se utiliza Dense Passage Retriever (DPR) (Karpukhin y cols., 2020) como modelo en la etapa de Retrieval. DPR utiliza

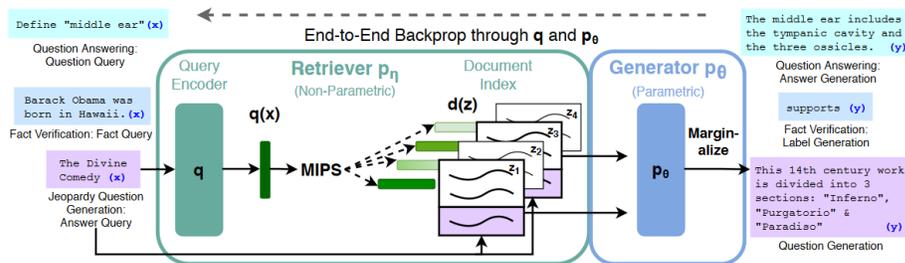


Figura 2.6: Método RAG utilizado en (P. Lewis, Perez, y cols., 2020). El retriever es un modelo DPR (véase sección 2.3.1) y el generator es un modelo de lenguaje. Figura extraída de P. Lewis, Perez, y cols. (2020).

una arquitectura *Bi-Encoder*. Primero, se usa un encoder $E_P(\cdot)$ que asocia cada documento z con un vector $v_z \in \mathbb{R}^d$. Al momento de procesar una pregunta q , se utiliza un encoder diferente $E_Q(\cdot)$ que asocia la pregunta con un vector $v_q \in \mathbb{R}^d$. Finalmente se obtienen los k documentos más cercanos según las distancias (o similitud) entre los vectores v_z asociados a los documentos y el vector v_q asociado a la pregunta. Se utiliza el producto punto para calcular la similitud entre vectores. El algoritmo para obtener los k documentos más cercanos se lo conoce como *k-Nearest Neighbors* (k-NN).

Se han propuesto otros métodos de generación de representaciones densas que podrían ser utilizados para este componente. En particular, existen varios trabajos que proponen modelos para la generación de *sentence embeddings* (embeddings de textos de largo arbitrario). A diferencia de DPR, estos modelos utilizan un solo encoder para generar los embeddings tanto de los documentos como de la pregunta o entrada. Se utiliza la similitud del coseno entre los embeddings de los documentos y de la pregunta para recuperar los k documentos más similares.

Un método comúnmente utilizado para la generación de *sentence embeddings* es usar la salida del primer token (CLS), o el promedio de la capa de salida, de un modelo de lenguaje enmascarado (por ejemplo, BERT (Devlin y cols., 2019)). El modelo Sentence-BERT (Reimers y Gurevych, 2019) mejora los resultados de estos embeddings modificando la arquitectura de BERT para incorporar la estructura de redes siamesas en la etapa de entrenamiento, como se observa en la Figura 2.7.

En L. Wang y cols. (2022) se presenta E5 (*EmbEddings from bidirectional Encoder rEpresentations*), con el objetivo de generar embeddings de propósito general. Para ello se construye un dataset a partir de datos sin etiquetar, lo que permite entrenar con una mayor cantidad de datos que en los métodos previamente mencionados.

Debido a la gran cantidad de modelos disponibles para generación de *sentence embeddings*, es necesario una manera de evaluarlos para las diferentes tareas

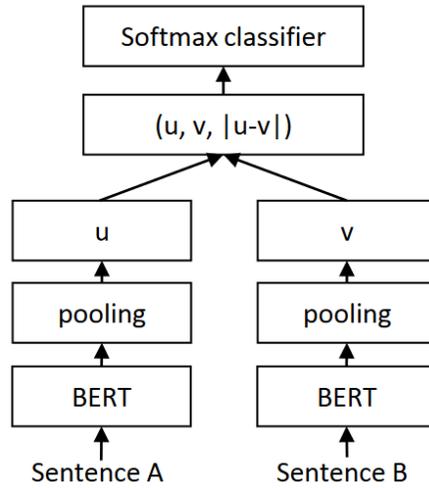


Figura 2.7: Arquitectura de sentence-BERT. Las dos redes BERT tienen los mismos pesos (redes siamesas). Obtenida de (Reimers y Gurevych, 2019).

que pueden ser utilizados. Por ello se construyó MTEB (*Massive Text Embedding Benchmark*) (Muennighoff, Tazi, Magne, y Reimers, 2022) que cubre 8 tareas en las que se utilizan embeddings, incluida la tarea de retriever. Actualmente se mantiene una tabla de clasificación utilizando dicho benchmark, accesible en *hugging face*³.

Si bien en P. Lewis, Perez, y cols. (2020) se utilizan representaciones vectoriales densas, obtenidas mediante modelos *Bi-Encoders*, es posible utilizar cualquier método de recuperación de información para la etapa de Retrieval. Estos métodos que utilizan modelos tipo *Bi-Encoder* para generar *sentence embeddings* son catalogados también con el nombre de *representation-based similarity* (similitud basada en representaciones) (Khattab y Zaharia, 2020), ya que la interacción entre la consulta y el *chunk* es a nivel de representación, y no hay interacción cruzada entre los tokens de ambas partes durante la generación de dichas representaciones. La Figura 2.8 (a) muestra este comportamiento de forma esquemática.

Una alternativa a usar representaciones vectoriales son los modelos *Cross-Encoder*, los cuales directamente procesan cada *chunk* con la consulta y retornan un score de similitud (Nogueira y Cho, 2019; Reimers y Gurevych, 2019). Estos modelos también se basan en la arquitectura BERT. Se entrenan tomando la concatenación de la consulta y el *chunk*, y se utiliza el vector CLS como entrada a una capa *feed-forward* que retorna la probabilidad de que el *chunk* sea relevante para la consulta (score de similitud).

Estos modelos son catalogados con el nombre de *all-to-all interaction* (in-

³<https://huggingface.co/spaces/mteb/leaderboard>

teracción de todos con todos) (Khattab y Zaharia, 2020), ya que, a diferencia de los modelos *Bi-Encoder*, la interacción entre la consulta y el *chunk* sucede a nivel de tokens en el procesamiento de las capas internas del modelo. Así, todos los tokens de la consulta y el *chunk* pueden interactuar de forma cruzada con el resto de tokens. En la Figura 2.8 (b) se muestra este comportamiento.

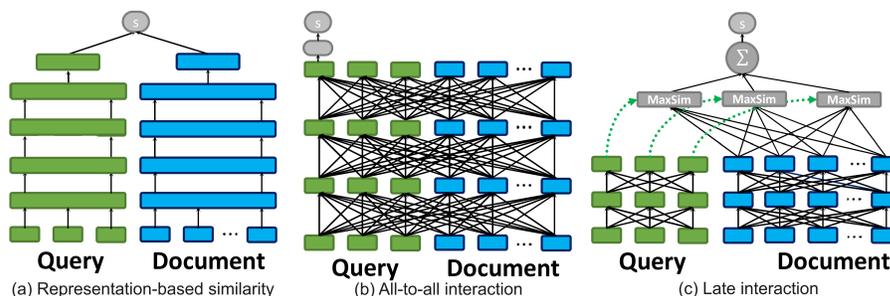


Figura 2.8: Comparación esquemática de las tres familias de métodos de Retrieval descriptas. Figura extraída de (Khattab y Zaharia, 2020).

Mientras que esta alternativa tiende a obtener mejores resultados, es significativamente menos eficiente. Suponiendo que se cuenta con n *chunks*, dada una consulta q , utilizando un modelo Cross-Encoder es necesario realizar n inferencias del modelo, una para cada *chunk*. En cambio, utilizando un modelo *Bi-Encoder*, se generan y almacenan los *sentence embeddings* para los *chunks* una vez, y dada una consulta q , solamente se realiza una inferencia del modelo para obtener el *sentence embedding* de la consulta. Luego, se deben realizar n cálculos de distancia (o utilizar algoritmos aproximados) que es mucho más eficiente que la ejecución de un modelo.

Como punto intermedio entre estas dos familias de modelos se propuso *ColBERT* (Khattab y Zaharia, 2020), un método que consiste en utilizar los embeddings contextualizados generados por un modelo tipo BERT para cada token, tanto de la consulta como del *chunk* (como en DPR, se utilizan modelos distintos para procesar la consulta y los *chunks*). A partir de estos embeddings, se obtiene la máxima similitud del coseno para cada token de la consulta con todos los tokens del *chunk* (operación llamada *MaxSim*), y se suman los resultados.

De esta forma, se mantiene la principal ventaja de los modelos *Bi-Encoder* ya que se pueden computar los embeddings de los *chunks* una vez y almacenarlos, y se mantiene cierta interacción cruzada entre los tokens de la consulta y el *chunk* al realizarse comparaciones cruzadas entre las representaciones de estos, similar a lo que sucede con los *Cross-Encoders*. A este tipo de método se le llama *late interaction*. La Figura 2.8 (c) resume este método de forma esquemática.

2.3.2. Ventana de contexto y RAG

En [P. Xu y cols. \(2024\)](#) se proponen investigar la efectividad de realizar *Retrieval Augmentation* en modelos con diferentes tamaños de ventana de contexto. Se busca responder si utilizar esta técnica tiene resultados competitivos con aprovechar las grandes ventanas de contexto para procesar todos los documentos juntos (sin etapa de Retrieval), y si combinar ambos métodos tiene mejor rendimiento que cada uno por separado.

Para llevar a cabo los experimentos, utilizan un modelo propietario de 43 mil millones de parámetros y **Llama 2** ([Touvron, Martin, y cols., 2023](#)) en su versión de 70 mil millones de parámetros, aumentando las ventanas de contexto de estos modelos de 4k a 16k, y en el caso de **Llama 2** también a 32k.

Experimentan con tres técnicas de retrievers diferentes y los extractos seleccionados con k -Nearest Neighbors son concatenados y utilizados directamente como entrada al modelo de lenguaje.

Los experimentos muestran que los modelos con ventana de 4k y sin retrieval obtienen los peores resultados. Esto es evidente porque el tamaño de la ventana no alcanza para ingresar todo el contexto y por ende se recorta de forma aleatoria. Se observa que utilizar *Retrieval Augmentation* ayuda considerablemente a los modelos con ventanas de 4k.

Otra observación interesante es que aún utilizando *Retrieval-Augmentation* con $k = 5$ como parámetro de k -Nearest Neighbors (en otras palabras, utilizando el mismo contexto como entrada para los distintos modelos), los modelos con ventanas de contexto más grandes obtienen mejores resultados. Esto se podría explicar por el fenómeno de “*lost in the middle*” ([Liu y cols., 2024](#)): los modelos tienden a prestar más atención a los tokens que se encuentran al principio y al final de la entrada, y menor atención a los tokens en el medio.

Se concluye que *Retrieval Augmentation* mejora de forma significativa el rendimiento de ambos modelos independientemente del tamaño de la ventana de contexto. Los mejores resultados se obtienen con el modelo de **Llama 2** con tamaño de ventana de 32k y *Retrieval Augmentation*, por lo que el uso del retriever es relevante aún con ventanas de contexto grandes. Además, los modelos con ventanas de 4k y *Retrieval Augmentation* son competitivos con los modelos de 16k que no utilizan esta técnica.

2.4. Técnicas de prompting

El trabajo que presentó al modelo **GPT-3** ([Brown y cols., 2020](#)) vino acompañado de un concepto novedoso emergente del uso de los LLMs llamado *In-Context Learning*.

In-Context Learning refiere a usar el texto de entrada de un modelo de lenguaje pre-entrenado para especificar un tarea. El modelo es condicionado por una instrucción en lenguaje natural y/o algunas demostraciones de la tarea (ejemplos) y se espera que complete nuevas instancias de la tarea simplemente prediciendo la continuación a dicha entrada, sin modificar de ninguna forma los



Figura 2.9: Ejemplo de *Zero-Shot*, *One-Shot* y *Few-Shot* aplicado a la tarea de traducción de inglés a francés. Figura traducida de [Brown y cols. \(2020\)](#).

pesos del modelo.

Se discriminan distintos tipos de *In-Context learning*:

- *Few-Shot* hace referencia a el escenario en donde el modelo recibe un conjunto chico de ejemplos de una tarea en tiempo de inferencia para condicionar su salida. Opcionalmente se cuenta con una descripción de la tarea al principio de la entrada.
- *One-Shot* hace referencia al caso particular de *Few-Shot* en donde solo se utiliza un ejemplo.
- *Zero-Shot* consiste únicamente de la descripción de la tarea en lenguaje natural, sin utilizar ningún ejemplo adicional.

La figura 2.9 muestra un ejemplo aplicando los diferentes tipos de *In-Context learning* a la tarea de traducción.

A partir de este trabajo y a medida que los LLMs fueron más accesibles, comenzaron a surgir varias técnicas novedosas para mejorar el rendimiento de estos modelos en tareas específicas utilizando únicamente el texto de entrada que recibe el modelo. Este texto de entrada se conoce generalmente como **prompt**, y en lo que sigue, utilizaremos este término para referirnos a él.

Una técnica de prompting que mejora el rendimiento de *Few-Shot* es *Chain-of-Thought* ([Wei y cols., 2024](#)). El objetivo de esta técnica es que el modelo genere una cadena de razonamiento, esto es, una serie coherente de pasos de razonamiento intermedios que conducen a la respuesta final de un problema.

Para lograr esto, se aplica *Few-Shot prompting*, y se agrega a cada ejemplo una cadena de razonamiento hasta el resultado esperado. La figura 2.10 muestra un ejemplo de *Chain-of-Thought prompting*.

El trabajo muestra que LLMs suficientemente grandes son capaces de generar cadenas de razonamiento si se presentan ejemplos siguiendo la técnica *Few-Shot*. Además, se presentan resultados para experimentos empíricos sobre razonamiento aritmético, razonamiento de sentido común y razonamiento simbólico.

Para el razonamiento aritmético, se concluye que *Chain-of-Thought* mejora los resultados respecto a *Few-Shot* pero solo con modelos que superan aproximadamente los 100 mil millones de parámetros. Sin embargo, en los escenarios

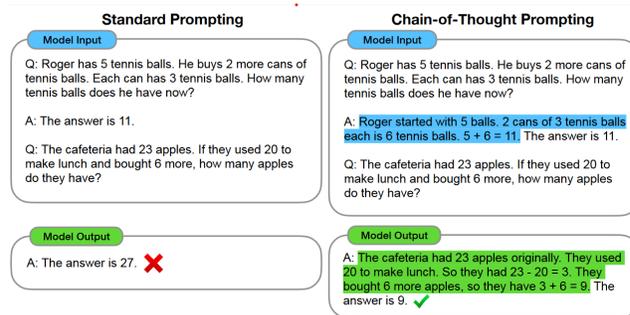


Figura 2.10: Ejemplo de un problema aritmético resuelto con *One-Shot* de forma incorrecta y resuelto con *Chain-of-Thought* de forma correcta. Figura obtenida de Wei y cols. (2024).

de razonamiento de sentido común y razonamiento simbólico, se observan mejoras para casi todos los *benchmarks* en todos los tamaños de modelos que se probaron.

X. Wang y cols. (2023) continúan el trabajo anterior introduciendo *Self Consistency prompting*. Esta técnica consiste en ejecutar *Chain-of-Thought prompting* múltiples veces sobre la misma entrada utilizando *sampling decoding*⁴, para generar un conjunto diverso de cadenas de razonamiento. Se elige como respuesta final el resultado que más veces se repite en todas las generaciones.

Se evalúa *Self-Consistency* en tareas de razonamiento aritmético, de sentido común y simbólico utilizando varios modelos de diferentes tamaños y se concluye que mejora a *Chain-of-Thought* con un margen alto en todas las tareas evaluadas.

Otra técnica que surge inspirada en *Chain-of-Thought* es *Tree-of-Thoughts* (Yao, Yu, y cols., 2023), un paradigma que permite a los modelos explorar varios caminos de razonamiento distintos. Para lograr esto, se interpreta al razonamiento como una búsqueda sobre un árbol. La motivación para este enfoque está en que métodos como *Chain-of-Thought* no permiten explorar diferentes continuaciones en una misma cadena de razonamiento, ni hacer *backtracking* cuando se llega a conclusiones equivocadas.

La figura 2.11 compara de forma esquemática las técnicas de *Zero-Shot*, *Chain-of-Thought*, *Self-Consistency* y *Tree-of-Thoughts*.

Se diseñan los siguientes pasos para seguir el enfoque de *Tree-of-Thoughts*:

- **Descomposición en pasos de razonamiento:** A diferencia de *Chain-of-Thought*, es necesario dividir la cadena de razonamiento en pasos, donde cada paso corresponderá a un nodo del árbol. Un paso de razonamiento puede ser una oración, una ecuación, un párrafo, etcétera.

⁴ *Sampling* es un método de decodificación (método para elegir el siguiente token dada la distribución de probabilidad sobre todos los tokens) que consiste en tomar el token aleatoriamente según la distribución de probabilidad $P(w_n|w_{1:n-1})$. Existen distintas variantes de *sampling*, como por ejemplo *random sampling*, *top-k sampling* y *nucleus sampling*.

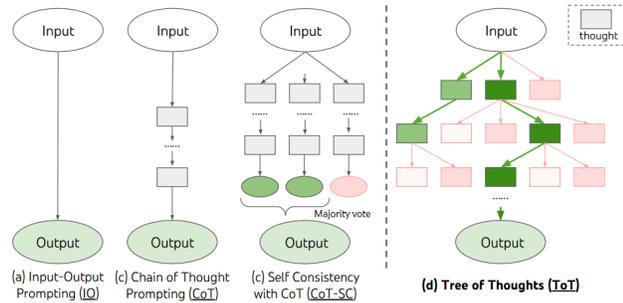


Figura 2.11: Ilustración esquemática donde cada rectángulo representa un paso de razonamiento. Se compara *Zero-Shot (IO)*, *Chain-of-Thought (CoT)*, *Self-Consistency (SC-CoT)* y *Tree-of-Thoughts (ToT)*. Figura obtenida de Yao, Yu, y cols. (2023).

- Generador de pasos de razonamiento:** Se deben generar k candidatos a pasos de razonamiento, a partir de un nodo padre, utilizando un LLM. Se presentan dos estrategias para generarlos: (1) Usar *Chain-of-Thought* prompting para generar los k candidatos utilizando una estrategia de *sampling decoding*, (2) generar los k candidatos utilizando una misma prompt, lo que permite evitar duplicados.
- Evaluador de estado:** Dados los k candidatos a paso de razonamiento, el evaluador de estado sirve para evaluar el progreso obtenido con cada candidato en pos de resolver el problema. Es necesario para determinar cuales estados seguir explorando y cuales no. Se proponen dos estrategias de evaluación utilizando LLMs: (1) Asignar un valor a cada candidato de forma independiente. El valor podría ser una etiqueta (por ejemplo: *sure/likely/impossible*) o un número (por ejemplo: 1-10). (2) Votar sobre los posibles candidatos utilizando una prompt que permita comparar todos los candidatos y elegir el más adecuado (o definir un orden).
- Algoritmo de búsqueda:** Se debe elegir un algoritmo de búsqueda para recorrer el árbol. En el trabajo se exploran los algoritmos *Breath-first search* y *Depth-first search*.

Se evalúa esta técnica para resolver el juego *Game of 24*, para escritura creativa y para resolver crucigramas 5x5. En estos escenarios se observa que *Tree-of-Thoughts* obtiene mejores resultados en distintas métricas que *Chain-of-Thought* y *Few-Shot*.

Otro trabajo que utiliza la idea de razonamiento es *React* (Yao, Zhao, y cols., 2023), en donde se busca integrar el razonamiento con la posibilidad de realizar acciones.

La técnica *React* busca generar pasos de razonamiento y acciones de manera intercalada para resolver una tarea particular. Esto permite generar planes

mediante razonamientos para ejecutar acciones (*reason to act*), así como interactuar con fuentes externas (por ejemplo Wikipedia) para incorporar información adicional a los razonamientos (*act to reason*).

Se genera una prompt utilizando *Few-Shot* para generar acciones (dentro de un conjunto predefinido) y razonamientos en lenguaje natural. Cada ejemplo utilizado en la prompt consiste en una secuencia de acciones, razonamientos y observaciones del ambiente, generada por un humano para resolver una instancia de un problema.

Se pretende que el modelo genere los pasos de razonamiento y las acciones. Estas últimas generan una observación del ambiente que es agregada a la prompt para que el modelo continúe la secuencia.

Los experimentos llevados a cabo incluyen utilizar la técnica mediante *Few-Shot prompting* así como realizar fine-tuning de modelos más pequeños para que aprendan el formato de las secuencias. Además, se experimenta con combinar *React* y *Self-Consistency (CoT-SC)* de dos maneras diferentes: (a) **React** \rightarrow **CoT-SC**: Cuando *React* no llega a un resultado en una cantidad fija de pasos, pasar a *CoT-SC*. (b) **CoT-SC** \rightarrow **React**: Cuando el resultado que ocurre mayoritariamente en las n generaciones de *CoT-SC* ocurre menos de $n/2$ veces, pasar a *React*.

Esta técnica de prompting es evaluada empíricamente en tareas de *question answering*, *fact verification*, el juego *ALFWorld* basado en textos y navegación de sitios web. Se concluye que la combinación de *React* y *CoT-SC* da los mejores resultados respecto a realizar solamente acciones o *CoT-SC*. Sin embargo, se observa que con *React* se obtienen mejores resultados utilizando fine-tuning en lugar de *Few-Shot prompting*.

2.4.1. Lógica y Chain-of-Thought

Otro problema que se ha observado en el uso de LLMs, es que suelen generar razonamientos erróneos (Valmeekam y cols., 2022). Algunos trabajos exploran la posibilidad de combinar lógica clásica con técnicas de prompting para mejorar este aspecto.

La técnica de prompting *Logical Thoughts (LoT)* (Zhao y cols., 2024) usa equivalencias lógicas, expresando premisas en múltiples formas lógicamente equivalentes para incentivar la exploración de diferentes soluciones. Esto se logra al incorporar un paso de verificación para cada razonamiento, donde una explicación es generada tanto para el razonamiento sin cambios, como para su versión lógicamente negada. El LLM es instruido a elegir entre las dos explicaciones.

Symbolic CoT (SymbCoT) (J. Xu y cols., 2024) es otro método propuesto que involucra cuatro módulos, todos utilizando LLMs para resolver las tareas:

1. **Translator**: Se traducen las premisas y la consulta a fórmulas en Lógica de Primer Orden.
2. **Planner**: Se divide el problema original en subproblemas más pequeños y se desarrolla un plan paso a paso.

3. **Solver:** Se deriva la respuesta a través de un proceso de inferencia lógica.
4. **Verifier:** Se valida la correctitud de las traducciones a fórmulas, y de la salida del *Solver*.

2.5. Métricas de evaluación

Originalmente, la manera de evaluar sistemas diseñados para la tarea de *Question Answering* consistía en utilizar métricas de *lexical matching* entre respuestas de un *gold standard* (provistas en un *benchmark*) y las predicciones del modelo. Sin embargo, a medida que se empieza a utilizar modelos generativos en lugar de extractivos, los métodos clásicos de evaluación empiezan a fallar. Esto sucede debido a que la lista de respuestas del *gold standard* es incompleta y no es posible cubrirlas todas. Además, los LLMs pueden generar respuestas con alucinaciones que parecen plausibles pero son incorrectas, haciendo aún más difícil la comparación (Kamalloo, Dziri, Clarke, y Rafiei, 2023).

En Kamalloo y cols. (2023), se evalúan 12 modelos de *open-domain question answering* con distintas estrategias para medir las discrepancias entre el rendimiento medido y el rendimiento real de estos modelos.

Una estrategia realizada es evaluar utilizando las métricas más comúnmente utilizadas de *lexical matching* que se definen en Rajpurkar, Zhang, Lopyrev, y Liang (2016) de la siguiente manera:

- **Exact-Match accuracy (EM):** La respuesta candidata es considerada correcta si se encuentra completamente en el conjunto de respuestas del *gold standard*. *EM accuracy* se calcula como el ratio de respuestas correctas en el conjunto de test.
- **F1 score:** Se calcula el F1 en función del solapamiento de los tokens de la respuesta candidata con los tokens de cada respuesta del *gold standard* (sin importar el orden). Se toma el F1 más alto entre el conjunto de respuestas del *gold standard* para cada respuesta candidata, y se calcula el promedio sobre todos los valores de F1.

Otra estrategia de evaluación investigada en Kamalloo y cols. (2023) es mediante similitud semántica, considerando la tarea de evaluación como un problema de clasificación donde el objetivo es decidir si la respuesta candidata y la respuesta de referencia son semánticamente similares o no. Se utiliza el modelo BEM (Bulian, Buck, Gajewski, Börschinger, y Schuster, 2022), entrenado siguiendo la arquitectura de BERT con un conjunto de datos especialmente construido para la tarea de *Answer Equivalence*.

También se utiliza como métrica de evaluación *Zero-Shot prompting* con un LLM. Se construye una prompt que contiene la pregunta, la respuesta candidata y la respuesta de referencia. Finalmente, se agrega la instrucción de determinar si la respuesta candidata es correcta.

Por último, se realiza evaluación humana para comparar todas estas estrategias de evaluación con el criterio humano. Algunos trabajos recurren directamente a evaluación humana para evaluar sistemas de QA. En [Min y cols. \(2021\)](#) se observa que el rendimiento de estos sistemas aumenta un 23% en promedio al ser evaluados por humanos, en comparación con la evaluación automática.

Luego de evaluar los 12 sistemas con estas estrategias, se observa que la correlación τ de Kendall entre la evaluación humana y la evaluación usando *Zero-Shot prompting* y BEM es de 0.75 y 0.70 respectivamente, mientras que EM y F1 tienen una correlación de 0.23 y 0.37 respectivamente. Esto muestra que las métricas que utilizan modelos de lenguaje (BEM y *Zero-Shot prompting*) se correlacionan bastante mejor con la evaluación humana que las métricas de *lexical matching*.

Se observa que los sistemas con LLMs son severamente subestimados por las métricas de *lexical matching*. De la evaluación humana se concluye que estos sistemas son los que producen los mejores resultados, y además son los que obtienen el mayor aumento respecto a las evaluaciones obtenidas con EM y F1.

De todas formas, se observa que los sistemas que usan LLMs tienden a ser evaluados de forma menos precisa por las estrategias *Zero-Shot* y BEM cuando son demasiado largas y tienen información inventada.

Otro trabajo que hace una valoración de distintas métricas es ([Yeh, Eskenazi, y Mehri, 2021](#)), centrándose en la evaluación para sistemas de diálogo, una tarea con muchos puntos en común con *question answering*. Se evalúan 23 métricas de evaluación diferentes en 10 datasets distintos.

Se clasifican estas métricas en función de tres aspectos: (1) Si usa un modelo pre-entrenado, (2) qué dataset se utiliza para entrenarla, y (3) si requiere una respuesta de referencia o no.

Se observa que las métricas basadas en reglas (BLEU [Papineni, Roukos, Ward, y Zhu \(2002\)](#), ROUGE [Lin \(2004\)](#) y METEOR [Banerjee y Lavie \(2005\)](#)) tienen problemas para evaluar respuestas generadas por los sistemas de diálogo que usan LLMs, debido a que se requiere un entendimiento semántico más profundo del contexto del diálogo y la respuesta generada.

Se concluye que USR ([Mehri y Eskenazi, 2020b](#)), GRADE ([Huang, Ye, Qin, Lin, y Liang, 2020](#)), DEB ([Sai, Mohankumar, Arora, y Khapra, 2020](#)), y USLH ([Phy, Zhao, y Aizawa, 2020](#)) son las métricas con mejor rendimiento para evaluar la generación de respuestas (*turn-level evaluation*).

Por otro lado, FlowScore ([Li, Zhang, Fei, Feng, y Zhou, 2021](#)), PredictiveEngage ([Ghazarian, Weischedel, Galstyan, y Peng, 2019](#)), DynaEval ([C. Zhang y cols., 2021](#)) y FED ([Mehri y Eskenazi, 2020a](#)) dan mejores resultados para evaluar el diálogo completo (*dialog-level evaluation*).

También se observa que las métricas presentan un mejor rendimiento cuando se evalúan con ejemplos provenientes del mismo dominio que los conjuntos de datos utilizados para su entrenamiento. Por lo tanto, se concluye que es importante buscar métodos para adaptar las métricas existentes a nuevos dominios.

2.5.1. LLMs como evaluadores (LLM-as-a-Judge)

En Zheng y cols. (2023) se plantea que los benchmarks tradicionales para evaluar los LLMs no representan correctamente las preferencias humanas ya que, por ejemplo, no muestran diferencias entre modelos base y modelos alineados mediante técnicas de fine-tuning (instruction tuning, RLHF, etc). Para hacer frente a esta problemática, los autores presentan dos benchmarks nuevos: *MTBench* y *Chatbot Arena*. *MTBench* se compone de una serie de preguntas abiertas que evalúan las habilidades de los modelos para mantener conversaciones multitermino y seguir instrucciones. *Chatbot Arena*⁵ es una plataforma que funciona mediante *crowdsourcing*, en donde las personas eligen entre dos modelos anónimos que son emparejados de forma aleatoria.

Por otro lado, en dicho trabajo se explora el uso de LLMs estado del arte como sustitutos a los humanos en el rol de evaluadores. A esto le llama *LLM-as-a-Judge*. Se realiza una comparación con un *gold standard* de evaluaciones humanas, para determinar el acuerdo entre anotadores al evaluar el benchmark *MTBench* así como un subconjunto de las preguntas realizadas por usuarios del *Chatbot Arena*.

Los autores muestran que hay un muy alto acuerdo entre GPT-4 y la evaluación humana. Incluso, se muestra que hay un mayor acuerdo entre GPT-4 y humanos (alcanzando un 85 %) que entre humanos (alcanzando un 81 %).

Una desventaja del uso de modelos como GPT-4 es que son cerrados y costosos, más aún cuando se trata de realizar múltiples evaluaciones sobre un conjunto grande de datos. *Prometheus* (Kim y cols., 2024) es un modelo *open-source* de 13 mil millones de parámetros, que fue entrenado específicamente para la tarea de evaluador. Para ello, los autores desarrollaron el *Feedback Collection Dataset*, que consiste de rúbricas de evaluación específicas, con instrucciones, y evaluaciones generadas con GPT-4, utilizando la técnica de Chain-of-Thought. El modelo *Prometheus* muestra una alta correlación con los humanos, a la par de la mostrada por GPT-4.

2.6. Síntesis

En este capítulo se presentó un resumen de algunos trabajos relacionados con los grandes modelos de lenguaje y la tarea de *generative question answering*.

Se observa que hay una tendencia a desarrollar técnicas que se integran con los LLMs para mejorar su rendimiento, con el fin de mitigar problemas como las alucinaciones o razonamientos erróneos. En este sentido, se destaca el método de *Retrieval-Augmented Generation* (RAG) que añade una etapa de recuperación de información con el fin de proveer el contexto necesario para generar respuestas más precisas.

Por otro lado, se observa que la evaluación automática de estos sistemas es un problema complejo, y aún no se ha encontrado una métrica que se asemeje a

⁵La plataforma está disponible en: <https://chat.lmsys.org/>, y posee un ranking de más de 100 modelos, determinado por más de 1.500.000 de votos anónimos.

la evaluación humana. Este será un desafío que se deberá abordar en el desarrollo de este trabajo.

En el siguiente capítulo se presenta una visión general de los métodos propuestos para este proyecto. Además, se describe la metodología empleada durante su desarrollo, se detalla la construcción del corpus de documentos y se presentan los modelos de lenguaje utilizados.

Capítulo 3

Búsqueda de respuestas en documentos institucionales

Los grandes modelos de lenguaje memorizan grandes cantidades de conocimiento en la etapa de entrenamiento, que les permite responder preguntas de dominio abierto de forma competente (Petroni y cols., 2019). Sin embargo, la información que se encuentra de forma mucho más esporádica en el conjunto de entrenamiento (llamada *long-tail knowledge*) no es capturada de la misma forma. Se ha mostrado que hay una relación entre la capacidad de los modelos de responder preguntas factoides y la cantidad de documentos asociados a la pregunta que se encuentran en el conjunto de entrenamiento (Kandpal, Deng, Roberts, Wallace, y Raffel, 2023). Por lo tanto, es de esperar que los modelos de lenguaje tengan dificultades para responder preguntas de dominios específicos, sobre los cuales es probable que haya muy poca información en el conjunto de entrenamiento.

Los LLMs presentan múltiples problemas que dificultan la resolución de la tarea de *question answering* para nuestro caso de estudio, como son:

1. **Alucinaciones:** Las alucinaciones refieren a respuestas generadas por los modelos que tienen información que no se desprende de los datos de entrenamiento, y por ende es inventada (Ji y cols., 2023). Además, los modelos no son capaces de reconocer si están alucinando o no, por lo que no son capaces de determinar si cuentan con el conocimiento para responder una pregunta.
2. **Razonamientos erróneos:** Los modelos de lenguaje tienden a generar razonamientos erróneos para tareas de mayor complejidad, lo que lleva a respuestas incorrectas (Valmeekam y cols., 2022).
3. **Falta de referencias:** Debido a que los LLMs son entrenados con un corpus que unifica texto de múltiples fuentes de información diferentes, no hay una manera de saber de qué fuente proviene la información que se utiliza para responder una pregunta.

4. **Desconocimiento de la información del dominio:** Como ya se explicó anteriormente, la información específica de un dominio particular no se suele encontrar bien representada en los corpus de entrenamiento, lo que hace imposible responder preguntas precisas sobre estos dominios.
5. **Frescura de la información:** Los LLMs son pre-entrenados con un corpus de texto, y luego son usados para inferencia en el mismo estado en que resultaron de la etapa de entrenamiento. Esto implica que el conocimiento del modelo no se actualiza, ni se retiene información ingresada durante el uso (son *stateless*).

Veamos estos problemas con algunos ejemplos concretos de preguntas que podrían surgir en nuestro caso de estudio. Las respuestas son generadas por el modelo **Claude 3 Haiku** de la empresa Anthropic (véase Sección 3.4 para más información del modelo). La única información que se ingresa como contexto al LLM es una breve prompt que indica el tópico de las preguntas (temas relacionados con la facultad de ingeniería), el estilo de las respuestas (cortas, en español), y se pide que se agregue una sección de referencias al final de la respuesta (prompt disponible en el Anexo A.1).

Pregunta
¿Cómo me inscribo a un curso?

Respuesta (claude-3-haiku-20240307)
Para inscribirse a un curso en la Facultad de Ingeniería de la Universidad de la República de Uruguay, debes seguir estos pasos:

1. Ingresar al sitio web de la Facultad de Ingeniería (<https://www.fing.edu.uy/>).
2. Buscar la **sección de "Estudiantes"** y luego la **opción de "Inscripciones"**.
3. Allí **encontrarás la información** sobre los períodos de inscripción y los requisitos necesarios.
4. Completar el **formulario de inscripción** con tus datos personales y la **información del curso que deseas cursar**.
5. **Enviar la solicitud de inscripción** siguiendo las instrucciones proporcionadas.

Una vez completado el proceso, recibirás la **confirmación de tu inscripción**.

****Referencias:****
- <https://www.fing.edu.uy/>
- <https://www.fing.edu.uy/estudiantes/inscripciones>

Figura 3.1: Ejemplo de respuesta a una pregunta del dominio usando el modelo de lenguaje **Claude 3 Haiku**. En rojo se resalta la información incorrecta y en verde la información correcta.

La Figura 3.1 presenta un ejemplo con la pregunta “¿Cómo me inscribo a un curso?”. Se observa que la respuesta comienza siendo bastante convincente, y se resalta en verde información que el modelo está generando (y que no es parte del contexto) que es correcta, como la URL al sitio web de la Facultad,

e incluso algunas secciones del sitio. Sin embargo, a medida que la respuesta avanza, el modelo comienza a cometer varios errores (marcados en rojo) de los mencionados anteriormente:

- Hay un **razonamiento erróneo** sobre la información presente en la pestaña *inscripciones* del sitio web, que refiere a inscripciones a carreras y no a cursos.
- El modelo **alucina** sobre la existencia de un procedimiento de inscripción a cursos que involucra un formulario de inscripción, lo cual no es correcto.
- Una de las **fuentes de información** ingresadas en la sección de referencias es un link inventado, que no dirige a ninguna página real dentro del sitio de facultad.

Pregunta

¿En qué fecha culminan las inscripciones a los cursos de 2024?

Respuesta (claude-3-haiku-20240307)

Según la información disponible en el sitio web de la Facultad de Ingeniería de la Universidad de la República de Uruguay, **las inscripciones a los cursos para el año 2024 culminan el 30 de noviembre de 2023.**

****Referencias:****

- <https://www.fing.edu.uy/inscripciones>

Figura 3.2: Ejemplo de respuesta a una pregunta del dominio usando el modelo de lenguaje Claude 3 Haiku. En rojo se resalta la información incorrecta.

La Figura 3.2 presenta otro ejemplo con la pregunta “¿En qué fecha culminan las inscripciones a los cursos de 2024?”. En este caso, se pueden resaltar varios problemas con la respuesta generada por el modelo:

- La pregunta requiere de **información actualizada** para ser respondida. Si bien el modelo hace referencia a supuesta información disponible en el sitio web, esto es una **alucinación** ya que no es capaz de acceder al sitio para extraer esa información.
- Este ejemplo evidencia el problema del **desconocimiento de información del dominio**, ya que la información requerida es suficientemente específica y particular, por lo que es seguro que no se encuentra bien representada en el corpus de entrenamiento del modelo.
- Nuevamente, la **referencia** es inventada.

Los ejemplos anteriores ilustran algunos de los problemas que se presentan al utilizar los LLMs de forma directa para responder preguntas específicas del caso de estudio (lo que se considera *Closed Generative QA*, sin información

de contexto). Para mitigar estos problemas, la estrategia que se sigue en este proyecto es proveer al modelo como contexto información relevante a la consulta (lo que se considera *Open Generative QA*).

Lo que sigue de este capítulo se compone de una visión general de los métodos explorados, y una descripción de la metodología de trabajo que se aplicó en el desarrollo de este proyecto. Finalmente, se presentan los pasos seguidos para la construcción del corpus de documentos y los LLMs utilizados.

3.1. Visión general de los métodos propuestos

En este proyecto se exploran distintos métodos para desarrollar un sistema de *Open Generative QA* sobre preguntas específicas del dominio de la Facultad de Ingeniería. Todos los métodos involucran el uso de LLMs para la generación de respuestas, pero difieren en la forma en que son utilizados, ya sea en la información que se provee al modelo como contexto, en la tarea que se pretende resolver o si se realiza fine-tuning del modelo o no. Los cuatro métodos propuestos, con sus diferencias, se presentan en la Tabla 3.1.

Método	Tarea del LLM	Retrieval	Entrenamiento
RAG simple	Responder una consulta utilizando la información del contexto	Sí	No
Fine-tuning de LLM	Responder una consulta utilizando la información del contexto	Sí	Sí
Prompting de derivaciones	Construir una derivación a partir de la información del contexto (véase Sección 4.3)	Sí	No
Grandes ventanas de contexto	Responder una consulta utilizando la información del contexto	No	No

Tabla 3.1: Diferencias entre los métodos propuestos. Se resaltan en rojo las particularidades de cada método respecto a RAG.

El método **RAG simple** consiste en realizar una etapa de recuperación de información para obtener los segmentos de texto relevantes a la consulta, y luego utilizar estos segmentos como contexto para generar la respuesta con un LLM, tal como se describió en el Capítulo 2 de Estado del Arte.

El método de **fine-tuning de LLM** consiste en realizar fine-tuning de un LLM *open-source* de menor tamaño, utilizando datos sintéticos generados con un LLM cerrado más grande.

El método de **prompting de derivaciones** es una propuesta original que consiste en construir la respuesta a la consulta a partir de la aplicación de reglas que combinan y/o transforman los *chunks* recuperados, construyendo un árbol al que llamamos derivación (inspirado en las derivaciones de lógica).

Por último, en el método de **grandes ventanas de contexto** se aprovecha el gran tamaño de las ventanas de contexto de los LLMs más recientes para utilizar los documentos completos como contexto, sin necesidad de realizar un proceso de recuperación de información.

Los métodos de fine-tuning de LLM y prompting de derivaciones pueden ser vistos como casos particulares del método RAG, ya que requieren de la etapa de Retrieval para recuperar los *chunks* relevantes a la consulta. En el método de fine-tuning, se utiliza el modelo resultante como LLM en la etapa de generación, mientras que en el método de prompting de derivaciones se busca resolver una tarea diferente que consiste en la construcción de la derivación.

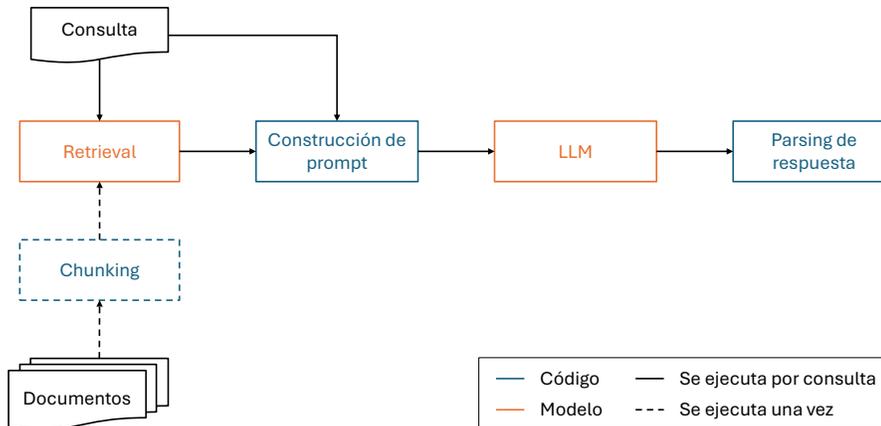


Figura 3.3: Esquema general de la arquitectura de los métodos propuestos.

La Figura 3.3 presenta un esquema general de la arquitectura que siguen estos métodos. Los módulos presentes en la arquitectura son los siguientes:

- **Chunking:** Módulo que se encarga de dividir los documentos en fragmentos más pequeños llamados *chunks*.
- **Retrieval:** Módulo que se encarga de recuperar los *chunks* relevantes a la consulta. Se utiliza un modelo *Bi-Encoder* con *k-Nearest Neighbors* o un modelo *Cross-Encoder*.
- **Construcción de prompt:** Módulo que se encarga de construir la prompt que se le ingresa al LLM, utilizando la información recuperada por el módulo de Retrieval.
- **LLM:** Modelo de lenguaje que se encarga de generar texto dada una prompt.
- **Parsing de respuesta:** Módulo que se encarga de procesar la respuesta generada por el LLM.

Cada método propuesto tiene sus particularidades. No todos los módulos son necesarios para todos los métodos, y en algunos casos se utilizan de forma diferente. Estas diferencias se presentan a continuación:

- El método de **Retrieval-Augmented Generation** no requiere del módulo de parsing de respuesta.
- El método de **fine-tuning de LLM** utiliza un modelo entrenado mediante fine-tuning con un formato especial de prompt y respuesta. Tampoco utiliza el módulo de parsing de respuesta.
- El método de **prompting de derivaciones** requiere del módulo de parsing de respuesta para construir la derivación a partir de la respuesta generada. Utiliza un formato de prompting especial para la generación de derivaciones, con *Few-shot prompting*.
- El método de **grandes ventanas de contexto** no requiere de los módulos de chunking ni Retrieval, los documentos van directamente al módulo de construcción de prompt. Tampoco requiere del módulo de parsing de respuesta.

3.2. Metodología de trabajo

A continuación se presenta la metodología de trabajo que se sigue en el desarrollo de este proyecto.

1. Proceso de experimentación

El primer paso en el desarrollo de este proyecto consiste en explorar las técnicas existentes, estudiadas en el documento de Estado del Arte (Anexo B), así como desarrollar nuevos métodos en un proceso de experimentación.

Este proceso de experimentación se realiza en un entorno controlado, utilizando un conjunto de documentos que se obtiene específicamente para este propósito. Todos los experimentos se realizan bajo las mismas condiciones, en pos de realizar una comparación justa entre los distintos métodos.

2. Metodología de evaluación

Se define una metodología de evaluación de los experimentos realizados, que permita comparar los distintos métodos propuestos. Para esto, se construye un dataset de preguntas y respuestas reales del dominio, que se utilizan como *gold standard* para evaluar las respuestas generadas con los distintos experimentos. Este conjunto es de evaluación y no es utilizado en la etapa anterior de desarrollo de los experimentos, para realizar una evaluación justa sobre un conjunto de datos no visto.

A medida que se desarrollan los experimentos, se evalúan utilizando la metodología definida y se registran los resultados obtenidos, para realizar un análisis posterior. Además, se realiza una evaluación humana sobre los experimentos con mejor rendimiento, para comparar los resultados obtenidos con la evaluación automática.

3. Desarrollo de un prototipo de la herramienta

Una vez que se determina cuál es el método que mejor se adapta para el contexto del problema a resolver, se procede a desarrollar un prototipo de un sistema de QA para responder consultas de estudiantes de Facultad de Ingeniería. Este prototipo tendrá como objetivo poder evaluar la utilidad potencial de la herramienta en un contexto de uso real, en un ambiente productivo.

4. Disponibilización del prototipo

Por último, el prototipo se pondrá a disposición para los integrantes del Equipo de Orientación y Consulta (EOC) para obtener retroalimentación. El *feedback* obtenido será utilizado para mejorar iterativamente la herramienta, con el objetivo de obtener un sistema que sea de utilidad para el EOC, y los estudiantes de la Facultad de Ingeniería.

3.3. Creación del corpus de trabajo

Para poder realizar una evaluación justa de cada método propuesto, se definió un conjunto de documentos extraídos de la página web de la facultad para ser utilizados como fuentes de información en todos los experimentos. Además se desarrolló un método para la generación manual de *chunks*: particiones de los documentos en bloques más chicos que serán utilizados en los experimentos con etapa de recuperación de información. A continuación se describen los pasos seguidos para la creación de este corpus de trabajo.

3.3.1. Selección y procesamiento de los documentos

Los documentos usados en la experimentación fueron obtenidos de la sección de enseñanza de la página web de la facultad: <https://www.fing.edu.uy/es/ensenanza>. Esta sección consiste de varias subsecciones con distinta información útil para el tipo de preguntas que se busca responder. En particular, se encuentra disponible toda la información del Espacio de Orientación y Consulta (EOC), quienes oficiaron de usuarios responsables del proyecto. También hay disponible información para la generación de ingreso, así como para futuros estudiantes, ambos grupos de personas que pueden necesitar este tipo de asistencia. Además, hay dos subsecciones de preguntas frecuentes, que pueden resultar especialmente útiles debido a que se adaptan al formato de nuestro problema (QA).

Mediante la técnica de scraping, se extrajeron 17 páginas web derivadas de la sección de enseñanza ya mencionada. Las páginas obtenidas en formato HTML fueron convertidas a formato Markdown utilizando la biblioteca `LangChain`, que provee la clase `Html2TextTransformer`. Esta clase ofrece `transform_documents`, una función que aplica una serie de reglas predefinidas para convertir los tags de HTML a la notación equivalente en Markdown. Cada página se guarda como un documento independiente. Además, se guardan algunos metadatos de cada página, como la URL y el título (extraído del HTML).

Luego, cada documento es procesado para eliminar información irrelevante, como el cabezal, menú, y pie de página. Debido a que todos los documentos fueron extraídos del mismo sitio, fue posible definir reglas para eliminar estas partes de los documentos de forma automática.

En la Tabla 3.2 se presenta la lista de documentos utilizados, junto con la URL de la página web de la que fueron extraídos y la cantidad de palabras que contiene cada documento.

La mayoría del contenido de los documentos extraídos es texto, pero también hay videos incrustados y enlaces a otras páginas. Se garantizó de forma manual que los enlaces estuvieran explícitamente escritos en el texto, para que el modelo de lenguaje pueda indicar los enlaces correctamente si es necesario. Los videos fueron sustituidos por el enlace al video, con una breve descripción del contenido del mismo.

La página titulada *Generación de ingreso* contiene una imagen con información relevante, que no es posible de extraer con la técnica de scraping utilizada. Para ese caso particular, se usaron las capacidades multimodales de *ChatGPT*¹, que permite procesar imágenes, para extraer la información relevante de la imagen y agregarla al documento.

Un documento particular que no fue utilizado en los experimentos, pero se procesó y agregó en el prototipo, es un PDF con información sobre los estudios preuniversitarios requeridos para ingresar a la Facultad de Ingeniería. Este documento tiene un formato particular, con tablas a lo largo de 15 páginas, y listas dentro de las tablas. Para procesar este documento, se convirtió cada página del PDF a una imagen, y se utilizó *ChatGPT* para extraer el texto de las imágenes, indicando que se transforme el formato de tablas y listas a un formato Markdown.

3.3.2. Generación de *chunks*

La generación de *chunks* a partir de los documentos es una tarea fundamental para la etapa de recuperación de información. Generar bloques de texto más acotados (*chunks*) que los documentos enteros tiene varias ventajas:

- **Tamaño de ventana de contexto de los modelos:** Tanto los modelos de lenguaje enmascarados que se utilizarán para las etapas de recuperación de información como los modelos de lenguaje generativos tienen ventanas de contexto de tamaño acotado. Esto significa que no todos los modelos permiten recibir como entrada documentos enteros. En estos casos, segmentar en *chunks* ya no es una ventaja sino una necesidad.
- **Coherencia temática de cada *chunk*:** Un mismo documento puede poseer información de diferentes temáticas. Estos casos pueden traer problemas tanto para su representación mediante *embeddings* como para la generación de texto. En el caso de los *embeddings*, debido a que son de

¹<https://chatgpt.com/>

Título	URL	# Palabras
Preguntas frecuentes	https://www.fing.edu.uy/es/ensenanza/generaci%C3%B3n-de-ingreso/talleres-de-orientaci%C3%B3n-al-inicio-toi	1426
Carreras de Grado	https://www.fing.edu.uy/es/ensenanza/carreras-de-grado-old	116
Espacio de Orientación y Consulta (EOC)	https://www.fing.edu.uy/es/ensenanza/espacio-de-orientaci%C3%B3n-y-consulta	463
Consultas y Orientación	https://www.fing.edu.uy/ensenanza/espacio-de-orientaci%C3%B3n-y-consulta/consultas-y-orientacion	123
Procedimientos para solicitudes	https://www.fing.edu.uy/ensenanza/espacio-de-orientaci%C3%B3n-y-consulta/procedimientos-para-solicitudes	840
Preguntas Frecuentes - EOC	https://www.fing.edu.uy/es/ensenanza/espacio-de-orientaci%C3%B3n-y-consulta/preguntas-frecuentes	324
Posgrado y Educación Permanente	https://www.fing.edu.uy/es/posgrado	107
¿Qué es un Posgrado?	https://www.fing.edu.uy/es/node/49534	447
Carreras de Posgrado	https://www.fing.edu.uy/es/ensenanza/carreras-de-posgrado	371
Cursos de Posgrado y Educación Permanente	https://www.fing.edu.uy/es/ensenanza/cursos-posgrado	288
GENERACIÓN DE INGRESO	https://www.fing.edu.uy/es/ensenanza/generacion-de-ingreso	137
INSCRIPCIÓN A LA FACULTAD	https://www.fing.edu.uy/node/45476	876
INSCRIPCIÓN A CURSOS	https://www.fing.edu.uy/node/45477	1319
ACTIVIDAD INTRODUCTORIA Y PRUEBA INICIAL	https://www.fing.edu.uy/node/45478	669
INICIO DE CURSOS	https://www.fing.edu.uy/node/45479	893
Subsidios a cursos de Educación Permanente en Fing	https://www.fing.edu.uy/es/node/47584	484
Futuros estudiantes	https://www.fing.edu.uy/es/node/41254	887

Tabla 3.2: Lista de URLs de páginas extraídas

largo fijo, cuanto mayor sea la diversidad semántica del texto a representar, menor será la precisión con la que sea posible capturar la semántica particular del documento. En el caso de la generación de texto, presentar información no relevante como contexto a la pregunta a responder podría dificultar la tarea de generación de una respuesta adecuada y relevante a la pregunta.

Existen múltiples maneras de generar los *chunks*. Una posibilidad es la generación automática mediante heurísticas a definir, como puede ser la definición de un largo máximo por *chunk* en cantidad de caracteres, palabras, oraciones, o párrafos. El problema con estas técnicas mediante reglas es que si la estructura

de los documentos no es muy predecible, es imposible garantizar la coherencia temática de cada *chunk*. Pueden haber bloques que comiencen descontextualizados, o que finalicen de manera abrupta.

Debido a esto, y siendo que la cantidad de documentos manejados es acotada (17 documentos), se decidió realizar esta tarea de forma manual. Para facilitar este proceso, se implementó una herramienta utilizando *React*. Esta herramienta permite visualizar un documento, seleccionar con el mouse las diferentes secciones del texto que componen a cada *chunk*, y guardar estas selecciones en un archivo CSV. La Figura 3.4 muestra la herramienta desarrollada en funcionamiento.

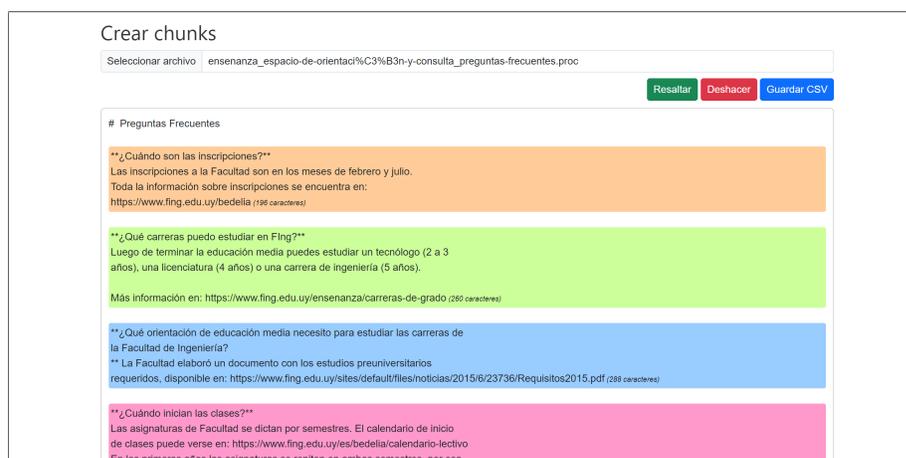


Figura 3.4: Ejemplo de uso de la herramienta para selección manual de *chunks* sobre un documento de preguntas frecuentes. Los distintos *chunks* seleccionados quedan marcados con diferentes colores.

Luego de procesar todos los documentos de esta manera, se generaron 96 *chunks* en total con 681 caracteres de largo en promedio. La Tabla 3.3 presenta algunos datos estadísticos de los *chunks* generados. Junto a cada *chunk* se almacena un identificador del documento original para no perder la trazabilidad.

3.4. Elección de modelos de lenguaje

Se decidió probar modelos de lenguaje abiertos y disponibles para descargar, así como modelos de lenguaje cerrados únicamente disponibles a través de APIs. Los modelos abiertos poseen varias ventajas por sobre los cerrados, principalmente:

- **Privacidad:** Es posible desplegar los modelos en una infraestructura nuestra, y por ende asegurar la privacidad de los datos enviados al modelo de lenguaje.

Métrica	Valor
Número total de chunks	96
Número total de documentos	17
Longitud de <i>chunks</i> (caracteres)	
Promedio	681
Mínimo	104
Máximo	1772
Desviación estándar	383
<i>Chunks</i> por documento	
Promedio	5.6
Mínimo	1
Máximo	15
Desviación estándar	4.3
Distribución de <i>chunks</i>	
Chunks con menos de 500 caracteres	43 %
Chunks entre 500 y 1000 caracteres	40 %
Chunks con más de 1000 caracteres	18 %

Tabla 3.3: Estadísticas detalladas de los *chunks* generados.

- **Control:** Si bien las distintas versiones de los modelos de lenguaje cerrados suelen estar identificadas y es posible a través de las APIs acceder a una versión específica, es imposible garantizar que los modelos no sean cambiados o actualizados, dificultando la evaluación de los experimentos y replicación de resultados.
- **Independencia:** La utilización de modelos de lenguaje cerrados a través de APIs genera una dependencia con los proveedores de las APIs. El uso de modelos abiertos desplegados en una infraestructura local es totalmente independiente de proveedores externos.

Sin embargo, también poseen múltiples desventajas importantes, principalmente:

- **Rendimiento:** Los grandes modelos de lenguaje de mayor rendimiento son entrenados por grandes empresas privadas, como OpenAI, Google, Anthropic o Meta, y en la mayoría de los casos se decide no dejarlos abiertos. Sin embargo, en el último tiempo se han puesto a disposición modelos abiertos que se acercan al rendimiento de los modelos cerrados más potentes, como la familia de modelos Llama 3 de Meta, o los modelos de Mistral. Aun así, los modelos con mejores resultados hoy en día siguen siendo cerrados (GPT 4o de OpenAI, Claude Sonnet 3.5 de Anthropic, Gemini 1.5 Pro de Google).

- **Requerimientos de hardware:** Otra importante desventaja del uso de modelos de lenguaje abiertos desplegados localmente es el costo computacional requerido para correr estos modelos. Los grandes modelos de lenguaje abiertos que se acercan al estado del arte requieren del uso de clusters de GPUs para poder ser ejecutados eficientemente, debido a su gran tamaño (por ejemplo, el modelo más grande de Llama 3 tiene 70 mil millones de parámetros). Esta limitante es crítica en el contexto en que se desarrolla este proyecto, ya que, debido a nuestras restricciones computacionales, solo fue posible experimentar con modelos más chicos, en torno a los 7 mil millones de parámetros. Estos modelos son aún inferiores en rendimiento a los modelos cerrados que son el estado del arte.

En función de las ventajas y desventajas presentadas, nos pareció oportuno realizar pruebas tanto con modelos abiertos como con modelos cerrados. Por un lado, resulta interesante evaluar qué tan cercanos son los resultados obtenidos con los modelos abiertos respecto a los obtenidos con los cerrados. Por otro lado, es también relevante determinar cuál es el mejor resultado posible a obtener usando los modelos que son el estado del arte.

La Tabla 3.4 presenta los modelos utilizados en los experimentos con una breve descripción de cada uno, la URL de descarga (en caso de ser abiertos) o documentación de la API (en caso de ser cerrados), y la referencia al trabajo que los presenta.

3.5. Síntesis

En este capítulo se presentó una visión general de los métodos que se proponen en este proyecto, así como la metodología de trabajo que se sigue. Se describió el proceso de creación del corpus de trabajo, la generación de *chunks* a partir de los documentos, y la elección de los modelos de lenguaje que se utilizan en los experimentos.

En el siguiente capítulo se aborda la experimentación realizada, con una descripción detallada de cada experimento y las decisiones tomadas en cada uno de ellos.

Modelo	Descripción	Abierto	URL	Referencia
Mistral 7B Instruct v0.2	Fine-tuning del modelo base Mistral 7B v0.2 para seguir instrucciones (<i>instruction fine-tuning</i>), provisto por Mistral.	Sí	https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2	(Jiang y cols., 2023)
OpenChat 3.5 0106	Fine-tuning del modelo base Mistral 7B v0.1 utilizando una técnica de reinforcement learning llamado C-RLFT.	Sí	https://huggingface.co/openchat/openchat-3.5-0106	(G. Wang y cols., 2024)
Llama 3 8B Instruct	Fine-tuning del modelo base Llama 3 8B utilizando las técnicas de fine-tuning supervisado y <i>reinforcement learning with human feedback</i> (RLHF), provisto por Meta.	Sí	https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct	(AI@Meta, 2024)
Claude 3 Haiku	Modelo más rápido y barato pero de menor rendimiento de la familia Claude 3 de Anthropic.	No	https://docs.anthropic.com/en/api/getting-started	(Anthropic, 2024)
Claude 3 Opus	Modelo de mejor rendimiento de la familia Claude 3 y competitivo con los modelos del estado del arte como GPT-4.	No	https://docs.anthropic.com/en/api/getting-started	(Anthropic, 2024)

Tabla 3.4: Descripción de Experimentos y Recursos Utilizados

Capítulo 4

Implementación de los métodos propuestos

En esta sección se explicarán los distintos experimentos desarrollados.

Cada experimento fue desarrollado en un Jupyter Notebook¹, con Python como lenguaje de programación, y se usaron los recursos del Centro Nacional de Supercomputación [Cluster.uy](https://cluster.uy) para las ejecuciones. En particular, el cluster cuenta con nodos que disponen de GPUs de alto rendimiento, como la GPU NVIDIA A100 (40 GB) y la GPU NVIDIA P100 (16 GB). Estos recursos son fundamentales para la ejecución eficiente de los modelos de lenguaje abiertos, así como la posibilidad de hacer fine-tuning.

Dependiendo de las necesidades del experimento, en algunos casos no fue necesario la utilización de GPUs, en otros alcanzó con el uso de un nodo con una GPU P100, y en otros fue necesario un nodo con una GPU A100. La Tabla 4.1 presenta un resumen de los experimentos realizados, con una breve descripción de cada uno, e información sobre los modelos y recursos utilizados.

Lo que resta de este capítulo se compone de una detallada explicación de cada experimento, sin entrar en la evaluación o resultados. El siguiente capítulo (Capítulo 5) abordará la forma de evaluación y los resultados obtenidos para cada experimento, así como un detallado análisis de estos resultados.

4.1. Retrieval Augmented Generation

El primer experimento que se llevó a cabo, y que sirve como base para otros experimentos que se presentan más adelante, es la implementación del método *Retrieval Augmented Generation* (RAG). Este método, como fue explicado en el Estado del Arte (Capítulo 2), consiste de dos etapas fundamentales: la etapa de recuperación (*retrieval*) y la etapa de generación (*generation*). La Figura 4.1

¹El código de los experimentos y del prototipo desarrollado quedará disponible en el siguiente repositorio: <https://github.com/nsuruguay05/fingchat>

Experimento	Descripción	Modelos	GPUs utilizados
RAG simple	Implementación del método Retrieval Augmented Generation (Sección 4.1)	Retrieval: Bi-Encoder o Cross-Encoder Generation: LLMs abiertos (Llama 3 8B, Mistral 7B, OpenChat 3.5) y cerrados (familia Claude 3)	Retrieval e inferencia: GPU NVIDIA P100
Fine-Tuning de LLM	Generación de dataset sintético usando modelo cerrado y Fine-Tuning de modelo abierto más chico (Sección 4.2)	Generación de dataset sintético: Modelo Bi-Encoder y Claude 3 Opus Fine-Tuning: Modelos abiertos (Llama 3 8B, Mistral 7B) Retrieval: Cross-Encoder	Fine-Tuning: GPU NVIDIA A100 Retrieval e inferencia: GPU NVIDIA P100
Prompting de derivaciones	Método de generación controlada mediante la aplicación de reglas para la construcción de derivaciones (Sección 4.3)	Retrieval: Bi-Encoder o Cross-Encoder Generation: Claude 3 Haiku y Claude 3 Opus	Retrieval: GPU NVIDIA P100
Grandes ventanas de contexto	Aprovechar modelos con grandes ventanas de contexto, que permiten agregar en el contexto todos los documentos completos (Sección 4.4)	Generation: Claude 3 Haiku y Claude 3 Opus	Ninguno

Tabla 4.1: Descripción de Experimentos y Recursos Utilizados

muestra un esquema de este método, donde se observa que la construcción de la prompt para el modelo generativo se realiza a partir de los *chunks* recuperados en la etapa de Retrieval. A continuación se explica más en profundidad las decisiones tomadas para cada componente.

4.1.1. Retrieval

La etapa de Retrieval consiste en recuperar los fragmentos de texto (*chunks*) más relevantes a la pregunta ingresada por el usuario. Se utilizan modelos de lenguaje enmascarados entrenados para esta tarea.

Se experimentó con dos familias de modelos para la recuperación de los extractos más relevantes para la pregunta: modelos *Bi-Encoder* y modelos *Cross-*

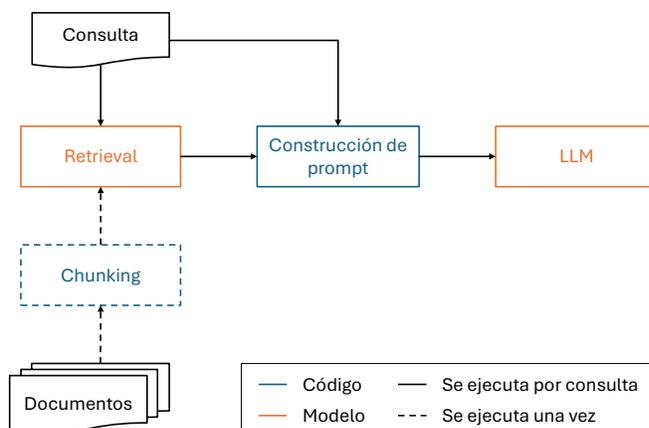


Figura 4.1: Esquema de funcionamiento del método RAG.

Encoder (explicados en el Capítulo 2 de Estado del Arte).

Los modelos *Bi-Encoder* se utilizan para la generación de *sentence embeddings*. Al utilizar estos modelos, se siguen dos etapas:

1. **Procesamiento de *chunks*:** Dada la colección de *chunks* obtenida a partir de los documentos, se genera una representación vectorial (*sentence embedding*) para cada uno de los *chunks*. Esta etapa se realiza una vez al procesar los documentos y los vectores son almacenados. A cada *chunk* se le agrega al principio el título del documento del que fue extraído, como información contextual adicional.
2. **Procesamiento de pregunta:** Al momento en que un usuario ingresa una pregunta al sistema, se genera la representación vectorial de la pregunta usando el mismo modelo que en la etapa anterior. Luego, se aplica la técnica de *k-Nearest Neighbors* para obtener los *k* vectores más cercanos al vector de la pregunta. Los *chunks* representados por estos vectores son la salida del componente de *Retrieval*.

Los modelos *Cross-Encoder*, en cambio, funcionan de una manera más directa: Al momento en que una pregunta es ingresada al sistema, se ejecuta el modelo tantas veces como cantidad de *chunks* haya, procesando cada *chunk* en conjunto con la pregunta y retornando un score de similitud. En este caso no se requiere de una etapa previa de generación de *embeddings* para los *chunks*, ya que el modelo *Cross-Encoder* retorna un score de similitud directamente, al procesar en conjunto la pregunta con cada *chunk*.

Los modelos *Cross-Encoder* tienden a dar mejores resultados (véase Capítulo 2), pero son bastante más ineficientes ya que se requieren múltiples inferencias por consulta, en contraste con los modelos *Bi-Encoder* que requieren solo una

inferencia para la generación del *embedding* de la consulta, y múltiples comparaciones de vectores. Aun así, debido a que en el contexto de este caso de estudio se tiene una cantidad pequeña de *chunks*, es posible utilizar modelos *Cross-Encoder* sin penalizar demasiado la eficiencia.

Se decidió evaluar ambos métodos, y en el Capítulo 5 se puede observar los resultados obtenidos con cada estrategia. El modelo *Bi-Encoder* que se utilizó en los experimentos es `intfloat/multilingual-e5-large` (L. Wang y cols., 2024), un modelo multilingüe entrenado por Microsoft. Se utiliza la biblioteca `SentenceTransformers` para la ejecución de este modelo y la implementación de *Nearest Neighbors* de la biblioteca `Scikit-Learn`. El modelo *Cross-Encoder* utilizado es `BAAI/bge-reranker-large` (Xiao y cols., 2024), un modelo entrenado especialmente para el inglés y chino, aunque se observó que se obtenían buenos resultados también en español (véase Capítulo 5). Luego de realizados los experimentos, se lanzó el modelo `BAAI/bge-m3`, una versión multilingüe del modelo, que no se llegó a poder utilizar. La biblioteca `SentenceTransformers` también es utilizada para la ejecución del modelo *Cross-Encoder*.

4.1.2. Generation

La etapa de generación consiste en la construcción de una *prompt* que toma los *chunks* obtenidos en la etapa de retrieval como contexto. Se utiliza esta *prompt* como entrada de un LLM.

Se diseñó la *prompt* con indicaciones generales de la respuesta esperada. En primer lugar se describe brevemente la tarea a realizar (responder preguntas sobre documentos de FING), y se especifica el estilo de respuesta esperado: respuestas cortas y concretas basadas únicamente en los extractos ofrecidos (los *chunks*). También se especifica que en caso de no encontrar la información para responder la pregunta en los extractos, se debe responder que no se cuenta con información para responder la pregunta. Por último se especifica el formato esperado para la respuesta, que debe contener al final una sección de referencias con los extractos que se decidan utilizar en la construcción de la respuesta.

Al momento de agregar los *chunks* a la *prompt*, también se agrega información adicional sobre cada *chunk*, como el título y la URL. Se puede observar la *prompt* utilizada en el Anexo A.2.

Se realizaron experimentos con todos los grandes modelos de lenguaje generativos que se presentan en la Tabla 3.4 (Sección 3.4), tanto abiertos como cerrados.

4.1.3. Cantidad de *chunks*

Un hiperparámetro fundamental en el esquema de RAG es la cantidad de *chunks* que son recuperados y agregados como contexto en la etapa de generación. Agregar una cantidad reducida puede llevar a dejar fuera información relevante a la pregunta, pero agregar demasiados puede derivar en agregar demasiada información irrelevante que conduzca al modelo a errores. Además, dependiendo del modelo utilizado y la capacidad computacional disponible, puede

haber una cota superior dada por el tamaño de la ventana de contexto o la memoria disponible (teniendo en cuenta que el costo de memoria es de orden cuadrático respecto al tamaño de la entrada).

Se evaluaron distintos valores para este hiperparámetro, tanto para el modelo *Mistral 7B Instruct v0.2* como para los modelos Claude 3, utilizando el *Cross-Encoder*. Los resultados pueden verse en el Capítulo 5.

4.2. Fine-Tuning de LLM

Este experimento se centra en la premisa de mejorar el rendimiento de los modelos de lenguaje abiertos que podemos ejecutar con nuestros recursos, usando los modelos de lenguaje cerrados, de mejor rendimiento, como “maestros”. Esta idea puede encontrarse en la literatura bajo el nombre de *Knowledge Distillation* (Hinton y cols., 2015; Beyrer y cols., 2022; Hsieh y cols., 2023).

Para esto, en esta experimentación nos proponemos generar un conjunto de preguntas y respuestas sintéticas basadas en los documentos utilizando el modelo de lenguaje “maestro”, y utilizar este conjunto para ajustar al modelo de lenguaje más chico, mediante el proceso de fine-tuning. Así, el modelo de lenguaje chico podría aprender los mecanismos y patrones utilizados en las respuestas generadas por el modelo “maestro”.

El modelo resultante del proceso de fine-tuning se utilizará como modelo de generación en el pipeline de RAG. Esto implica que el modelo aún necesitará recibir los *chunks* más similares en el contexto. La Figura 4.2 muestra un esquema de este proceso.

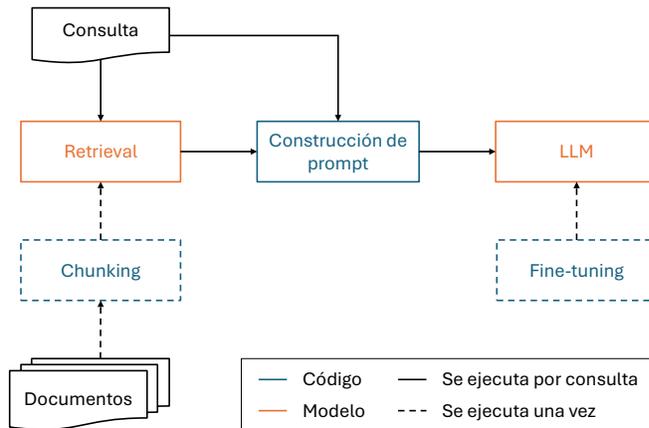


Figura 4.2: Esquema de funcionamiento del método de Fine-Tuning de LLM.

El proceso de fine-tuning no tiene como objetivo que el modelo aprenda sobre el contenido de los documentos. En cambio, se espera que el modelo mejore su

rendimiento en la tarea de generación de respuesta a partir de los *chunks* del contexto, en el dominio específico en que está enmarcado. También se espera que el modelo capture las particularidades del lenguaje utilizado en este dominio específico.

Es importante notar que no se espera que este experimento mejore los resultados obtenidos mediante la aplicación de RAG utilizando el modelo “maestro”, sino que lo que se busca es acercar el rendimiento del modelo más chico (y abierto, con las ventajas que esto trae) al rendimiento del modelo más grande.

A continuación se explicarán los dos procesos fundamentales de este experimento: La generación del conjunto de datos sintético y el fine-tuning del modelo de lenguaje.

4.2.1. Generación de conjunto de datos sintético

El objetivo es generar un conjunto de tuplas $\langle P, R, \{c_1, c_2, c_3\} \rangle$, en donde P es una pregunta, R es una posible respuesta para P y c_i son los *chunks* a partir de los cuales la respuesta R debe ser generada.

Para lograr esto, se diseñó un pipeline que consiste en dos pasos principales: (1) selección de *chunks* aleatorios, (2) generación de pregunta y respuesta. Además se generaron preguntas fuera del dominio para las cuales el modelo debería responder que no tiene información. Esto se explica en detalle más adelante. La Figura 4.3 muestra un esquema del proceso de construcción del dataset sintético.

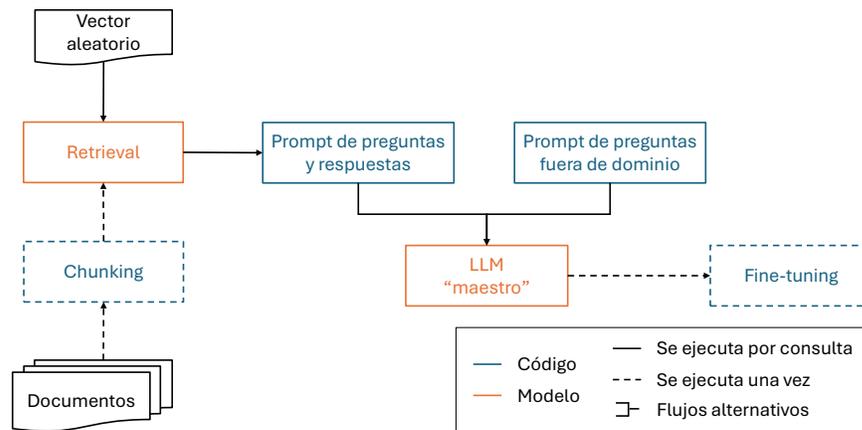


Figura 4.3: Esquema de construcción del dataset sintético para el fine-tuning de LLM.

Selección de *chunks* aleatorios

Para generar tanto la pregunta como la respuesta, es necesario primero seleccionar los *chunks* que deberían ser utilizados a la hora de responder dicha pregunta. La dificultad de este proceso radica en que la pregunta aún no existe, por lo que la selección de los *chunks* debe ser aleatoria.

Una primera alternativa utilizada fue sortear tres *chunks* de forma aleatoria, para ser utilizados luego en la generación de la pregunta y la respuesta. El problema de esta alternativa es que los *chunks* seleccionados no necesariamente están relacionados entre sí, por lo que la respuesta generada estará probablemente basada en uno solo de los *chunks* seleccionados. Sin embargo, hay múltiples escenarios en donde es deseable que el modelo aprenda a extraer información de múltiples *chunks*.

Para solventar este problema, se diseñó un algoritmo que consiste en:

1. Generar un vector aleatorio v (normalizado) de dimensiones iguales a los *embeddings* generados para los *chunks* para la etapa de Retrieval en RAG.
2. Tomar los tres *sentence embeddings* más cercanos al vector v utilizando *k-Nearest Neighbors*.
3. Los *chunks* correspondientes a los *sentence embeddings* más cercanos son seleccionados.

La intuición es que los *chunks* resultantes deberán estar relacionados entre sí según algún aspecto artificial que es capturado por el vector construido aleatoriamente.

Generación de pregunta y respuesta

A partir de los *chunks* ya seleccionados en el paso anterior, se debe generar una pregunta que pueda ser respondida utilizando al menos uno de estos *chunks* y la respuesta que efectivamente responda a dicha pregunta utilizando la información de los *chunks*. Este paso utiliza el modelo de lenguaje “maestro” para la generación de las preguntas sintéticas y las respuestas.

Se diseñaron dos prompts (véase Anexo A.3) con pequeñas diferencias en el enfoque de la tarea a resolver:

1. **Utilizar *chunks* necesarios:** En este enfoque la tarea a resolver es generar una pregunta y la respuesta a dicha pregunta, en donde sea necesario utilizar al menos uno de los *chunks* obtenidos del paso anterior para responder la pregunta. El modelo también tiene la responsabilidad de indicar cuáles son estos *chunks* (dentro de los tres dados por el paso anterior).
2. **Utilizar todos los *chunks*:** En contraste con el enfoque anterior, en este se espera que el modelo genere preguntas y respuestas para las que sea necesario utilizar todos los *chunks* provistos en el contexto. Para este enfoque no siempre se utilizan los tres *chunks* obtenidos del paso anterior,

sino que se elige aleatoriamente la cantidad de *chunks* (entre 1 y 3) que se pasan como contexto.

Ambos enfoques tienen dificultades particulares que el modelo debe resolver. Mientras que en el primer enfoque se espera que el modelo indique explícitamente los *chunks* necesarios para responder la pregunta, en el segundo enfoque se espera que el modelo genere correctamente preguntas (y sus respectivas respuestas) que realmente necesiten información de todos los *chunks* para ser respondidas correctamente.

En ambos casos la *prompt* consiste de tres partes: (1) descripción de la tarea a resolver, (2) listado de *chunks* a utilizar, (3) formato esperado de la salida. La tercera parte es fundamental para poder hacer parsing luego del texto generado y extraer correctamente la pregunta, la respuesta, y, en el caso del primer enfoque, las *chunks* utilizados.

Una vez obtenida la pregunta, respuesta y *chunks* utilizados, se debe generar la tupla $\langle P, R, \{c_1, c_2, c_3\} \rangle$ que corresponde a una instancia del conjunto de datos sintético. La pregunta P se corresponde directamente a la pregunta generada. La respuesta R es el string de la respuesta generada, al que se le agrega al final una sección “**Referencias:**” en donde se listan los nombres con las URLs de los documentos que corresponden a los *chunks* que fueron efectivamente utilizados en la generación de la respuesta. Por último, los tres *chunks* obtenidos del primer paso componen el conjunto de *chunks* de la tupla (esto incluye los *chunks* no utilizados en la construcción de la respuesta).

Además, para aumentar la diversidad en las preguntas generadas y que se parezcan más a situaciones reales, se experimentó agregando a las prompts una frase indicando el estilo de la pregunta esperada. Esta frase comienza con: “La consulta debe ” y es completada eligiendo de forma aleatoria (y equiprobable) entre los siguiente estilos: “ser una pregunta”, “estar escrita con un lenguaje informal”, “estar escrita como una frase, sin signos de interrogación”, “ser un párrafo largo explicando una situación particular del estudiante”, “ser corta e imprecisa”.

Se generaron 1192 tuplas siguiendo este método, algunas mediante el primer enfoque de prompt y otras utilizando el segundo enfoque de prompt, para tener mayor variedad de generaciones.

Generación de preguntas fuera del dominio

Uno de los aspectos para el cual el sistema a diseñar debe ser robusto es no responder preguntas fuera del dominio. Para estas preguntas no relacionadas, se pretende que la respuesta generada sea que no se cuenta con información para responder la pregunta. Podemos categorizar las posibles preguntas de la siguiente manera:

- **Preguntas fuera del dominio:** Corresponde a las preguntas que no están relacionadas con el dominio particular. En nuestro caso serían preguntas no relacionadas con aspectos administrativos de la Facultad de Ingeniería.

- **Preguntas del dominio sin información:** Corresponde a las preguntas que están relacionadas con el dominio, pero para las cuales no se cuenta con información en los documentos para responderlas correctamente.
- **Preguntas del dominio con información:** Corresponde a las preguntas que están relacionadas con el dominio y además se cuenta con información en los documentos para ser respondidas correctamente.

La única categoría de preguntas para las cuales se pretende que el sistema genere una respuesta que intente responder la pregunta es la última (preguntas del dominio con información). Para las otras dos categorías, el sistema debería responder que no cuenta con la información para responderlas, aún en el caso en que podría responderlas correctamente usando el conocimiento adquirido en la etapa de pre-entrenamiento del modelo generativo.

Dentro de las dos categorías de preguntas sin información, las preguntas fuera de dominio son bastante más sencillas de generar sintéticamente que las otras. Para las preguntas del dominio sin información, sería necesario procesar todos los documentos en conjunto para garantizar la generación de preguntas que no puedan ser respondidas con los documentos. En cambio, las preguntas fuera de dominio se pueden generar de forma sencilla mediante prompting, y podemos agregarlas en el conjunto de datos sintéticos.

Para generar estas preguntas, se diseñó una sencilla prompt aplicando Few-Shot prompting (véase Anexo A.3). Luego, la tupla agregada está formada por la pregunta generada, la respuesta: “Lo siento, pero no puedo ayudarte con eso ya que soy un modelo de lenguaje entrenado para responder preguntas relacionadas con la Facultad de Ingeniería de la Universidad de la República, de Uruguay.”, y se toman los tres *chunks* más cercanos a la pregunta como el conjunto de *chunks*.

Mediante este método, se aumentó el dataset sintético con 150 preguntas fuera del dominio, dando un total de 1342 tuplas que componen el dataset sintético generado.

Clasificador de preguntas fuera del dominio

Si bien el objetivo de este dataset sintético es hacer fine-tuning de un modelo de lenguaje generativo, se exploró la posibilidad de utilizar este conjunto para entrenar clasificadores de preguntas entre la categoría de preguntas fuera del dominio, y las otras dos categorías de preguntas relacionadas con el dominio. Un clasificador efectivo en esta tarea podría ser utilizado en el inicio del pipeline del sistema. Si la pregunta es clasificada como fuera del dominio, una respuesta automática puede ser desplegada y la pregunta nunca es procesada por el resto del sistema. Esto evitaría tener que cubrir este tipo de casos a la hora de generar la respuesta, que podría llevar a alucinaciones, generación de respuestas no deseadas (sobre temas delicados), o abusos de uso (usuarios que pretendan utilizar la herramienta como un generador de texto libre, estilo ChatGPT).

En primer lugar se hizo un análisis del dataset generado mediante la aplicación del algoritmo de clustering *K-Means*. Todas las preguntas fueron converti-

das a *sentence embeddings* de la misma forma que en la sección 4.1.1. Se aplicó el algoritmo *K-Means* sobre estas representaciones vectoriales con $k = 2$, con el objetivo de determinar si los clusters resultantes coinciden o no con las dos familias de preguntas. Se aplicó el algoritmo PCA (*Principal Component Analysis*) para reducir las representaciones a vectores de dos dimensiones y graficarlos.

La Figura 4.4 presenta dicha gráfica. En azul se observan las preguntas relacionadas con el dominio, que además fueron agrupadas en el mismo cluster. En amarillo, se observan las preguntas fuera de contexto agrupadas en el otro cluster. El punto negro corresponde a la única pregunta que no fue correctamente agrupada en el mismo cluster que el resto de preguntas de su categoría. Es una pregunta fuera del dominio que fue incluida en el cluster que contiene las preguntas relacionadas con el dominio.

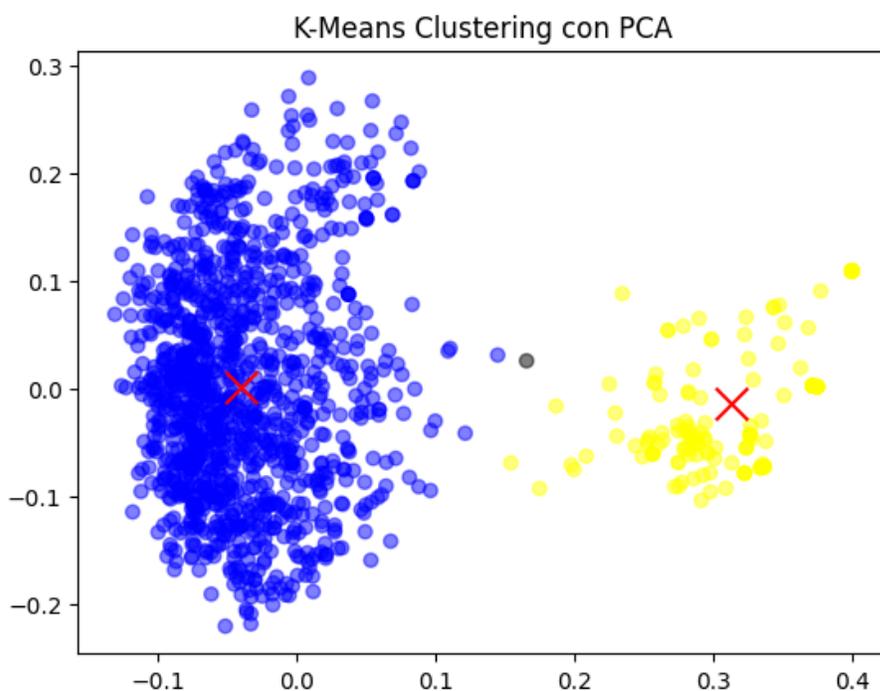


Figura 4.4: Gráfica de las representaciones vectoriales de las preguntas generadas aplicando PCA.

Como se puede observar, la separación generada por el algoritmo *K-Means* es casi perfecta, y en la gráfica son claramente diferenciables los dos clusters. Esto puede servir como indicación de que la representación semántica de los *sentence embeddings* captura correctamente las diferencias entre las preguntas, y podrían ser utilizados en algoritmos de clasificación.

A partir de esta idea, se experimentó con métodos clásicos de aprendizaje automático, como regresión logística, SVM y k -Nearest Neighbors. Se utiliza el conjunto de *sentence embeddings* de las preguntas del dataset sintético como datos de entrenamiento, y se clasifica la nueva pregunta a partir de su *sentence embedding*. Se probó a utilizar el dataset sintético completo (donde solo un 11 % de las preguntas son fuera del dominio) y a balancear el dataset eliminando preguntas del dominio para que quede perfectamente balanceado (150 preguntas del dominio y 150 preguntas fuera del dominio).

Se evaluaron estos métodos con datasets reales de preguntas relacionadas y no relacionadas obteniendo resultados muy prometedores, como se explica en el Capítulo 5.

En resumen, este método permite construir un clasificador de preguntas fuera del dominio de forma no supervisada (ya que la construcción del dataset sintético no requiere datos anotados), tomando como única entrada los documentos a usar para las respuestas (para la generación del dataset sintético). Este es un resultado directo de la construcción del dataset sintético, más allá del experimento principal para el que fue construido, el cual se sigue explicando en la siguiente sección.

4.2.2. Fine-tuning del modelo de lenguaje

Luego de generado el dataset sintético, el siguiente paso en el experimento consiste en realizar fine-tuning de modelos de lenguaje abiertos de menor tamaño utilizando estos datos. Para realizar el fine-tuning, dadas las restricciones de cómputo, se utilizará el método de *Low Rank Adaptation* (LoRA), explicado en el Capítulo 2.

Para ajustar el modelo de lenguaje generativo, es necesario convertir las tuplas $\langle P, R, \{c_1, c_2, c_3\} \rangle$ del dataset en entradas de texto con la prompt y la salida esperada. Luego, el modelo es entrenado de forma auto supervisada sobre estas entradas. Debido a que el objetivo es utilizar el modelo resultante en el pipeline de RAG, estas entradas de texto consisten en exactamente la misma prompt diseñada para el experimento de RAG (en Anexo A.2), agregando los tres *chunks* de la tupla en el contexto, así como la pregunta P y la respuesta esperada R .

Se utiliza la implementación de LoRA disponible en la biblioteca PEFT, cargando el modelo cuantizado en 4 bits para minimizar la cantidad de memoria del GPU necesaria. Se utiliza un rango $r = 16$, dropout de 0.1 y $\alpha = 32$. Se ajustan únicamente las matrices Q (query) y V (value) de las capas de atención del modelo. El entrenamiento se realiza durante 5 épocas con learning rate de 0.0002.

Se hizo fine-tuning de los modelos Mistral 7B Instruct v0.2 y Llama 3 8B (modelo base), y se evaluaron dentro del pipeline de RAG, como se explica en el Capítulo 5. Las Figuras 4.5 y 4.6 muestran la evolución de la función de pérdida a lo largo de los *steps* (320 por época) para cada uno de los modelos entrenados.

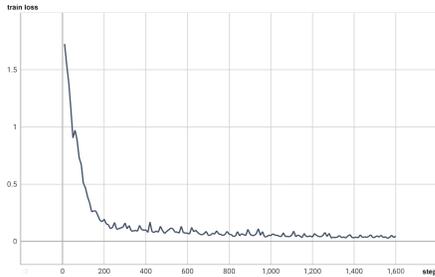


Figura 4.5: Evolución de la función de pérdida a lo largo de los *steps* en el proceso de fine-tuning del modelo Mistral.

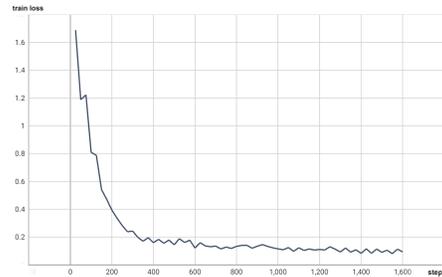


Figura 4.6: Evolución de la función de pérdida a lo largo de los *steps* en el proceso de fine-tuning del modelo Llama 3.

4.3. Prompting de derivaciones

Como se explicó en detalle en el Capítulo 2 del Estado del Arte, múltiples trabajos presentan alternativas para mejorar la capacidad de razonamiento de los LLMs. Por ejemplo, técnicas como Chain-of-Thought (CoT) (Wei y cols., 2024) han ayudado a mejorar el rendimiento de los modelos en varias tareas, incluyendo QA. Sin embargo, estas técnicas no definen explícitamente la forma en que el modelo debe razonar, ya que no se definen restricciones en la manera en que los pasos de razonamiento intermedios deberían ser construidos.

Con la motivación de mitigar esta limitación presente en este tipo de técnicas, se desarrolló un nuevo método de prompting al que llamamos **derivaciones**, debido a que está inspirado en las derivaciones lógicas. Este método se propone como un enfoque alternativo para el paso de generación en el esquema RAG, y consiste en inferir una conclusión (respuesta) a partir de un conjunto de hipótesis iniciales (*chunks*), mediante la aplicación de reglas bien definidas que permiten transformar y/o combinar estas hipótesis. La Figura 4.7 muestra un esquema de este método, en donde se observa que se agrega un paso adicional de post-procesamiento luego de la generación de la respuesta, en el que se construye la derivación y se extrae la respuesta.

Este nuevo enfoque presenta algunas ventajas frente a otras técnicas existentes, principalmente:

- **Interpretabilidad:** Además de generar una respuesta final, este método produce también una estructura arborescente, a la que llamamos derivación (véase Figura 4.8). Cada nodo de la derivación representa la aplicación de una regla, la cual realiza cierta transformación a los nodos hijos. Esta estructura permite identificar de manera sencilla errores que el modelo pueda haber hecho, así como entender el camino llevado a cabo para llegar a la respuesta final.
- **Generación controlada:** Debido a que la respuesta se genera a partir

de la aplicación secuencial de reglas predefinidas, este método provee una guía estructurada que el modelo está condicionado a seguir. Este control sobre la generación permitiría reducir alucinaciones y razonamientos incorrectos, y ayuda a mantener la respuesta acotada a la información de los documentos.

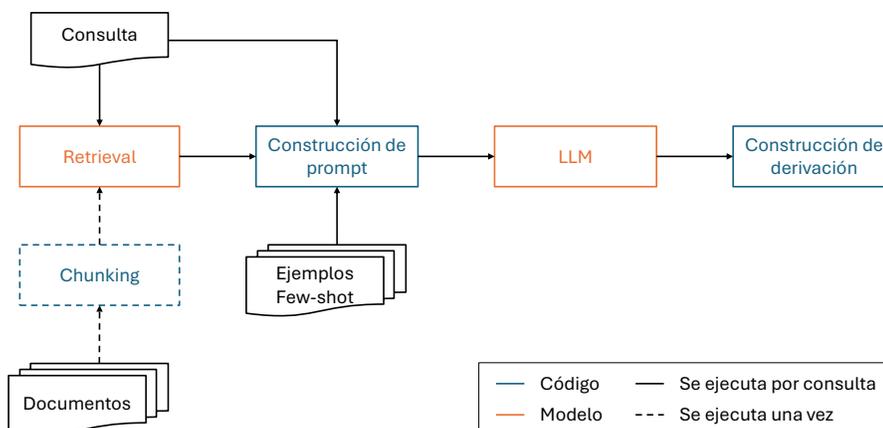


Figura 4.7: Esquema de funcionamiento del método de prompting de derivaciones.



Figura 4.8: Ilustración esquemática de una derivación generada usando prompting de derivaciones.

4.3.1. Definición del método

Esta técnica se centra en el paso de generación de RAG, y parte de la siguiente premisa: *La respuesta esperada para una consulta de usuario debe ser obtenida combinando y/o transformando la información más relevante extraída de la base de documentos.* Esto se enmarca en que el objetivo es utilizar únicamente la información disponible en los documentos, en contraposición con la información que el modelo pueda haber aprendido en la etapa de entrenamiento (i.e. memoria paramétrica).

La idea de este método está inspirado en como se construye un árbol de derivación en lógica proposicional. En el contexto de las derivaciones lógicas, decimos que una conclusión φ se deriva de un conjunto de premisas o hipótesis $\Gamma = \{\delta_1, \dots, \delta_n\}$, y lo notamos $\Gamma \vdash \varphi$, si dicha derivación existe. Definimos a esta clase de árboles como un conjunto definido inductivamente, en donde las reglas de la definición del conjunto determinan cómo derivar nuevas conclusiones a partir de derivaciones ya construidas. Los elementos (árboles) que pertenecen al conjunto de derivaciones se componen de nodos internos que representan la aplicación de las reglas, hojas que representan las hipótesis, y raíz que representa la conclusión (Van Dalen, 2013). La Figura 4.9 muestra un ejemplo de una derivación de lógica proposicional.

$$\frac{\frac{p_1 \wedge p_2}{p_2} E\wedge \quad \frac{p_1 \wedge p_2}{p_1} E\wedge}{p_2 \wedge p_1} I\wedge$$

Figura 4.9: Ejemplo de derivación que prueba $p_1 \wedge p_2 \vdash p_2 \wedge p_1$, donde p_1, p_2 son letras proposicionales, y $E\wedge, I\wedge$ son las reglas de eliminación e introducción del \wedge (respectivamente), según se define en Van Dalen (2013).

Siguiendo la analogía con las derivaciones de lógica, en prompting de derivaciones, se consideran los *chunks* más relevantes (obtenidos siguiendo las mismas estrategias que en la Sección 4.1.1) como el conjunto de hipótesis (o premisas) $\{h_1, \dots, h_n\}$. El objetivo es construir un árbol de derivación usando reglas predefinidas en lenguaje natural, hasta obtener una conclusión c , tal que $h_1, \dots, h_n \vdash c$, como se muestra en la Figura 4.8.

Mientras que en las derivaciones lógicas usualmente se parte de una conclusión candidata y se busca construir la prueba de dicha conclusión, en el caso de prompting de derivaciones la conclusión no se conoce de antemano. Entonces, la consulta q es necesaria para guiar la construcción del árbol, con el objetivo de que la conclusión resultante cumpla el rol de respuesta a la consulta q .

En cada paso de construcción de la derivación, la tarea que el modelo de lenguaje debe resolver consiste en decidir qué regla aplicar, seleccionar las hipótesis apropiadas, y generar la conclusión que surge de la aplicación de la regla elegida con dichas hipótesis. Es importante notar que este método se basa en la capacidad del modelo de lenguaje para desambiguar el lenguaje natural, tanto en las definiciones de las reglas como en las hipótesis, lo que le permite determinar qué regla aplicar.

4.3.2. Reglas

Se debe definir un conjunto de reglas a aplicar en cada paso de la construcción de la derivación. Estas reglas se especifican en lenguaje natural y son utilizadas por el LLM para construir el árbol.

Se define un conjunto de reglas de derivación que son convenientes para

nuestro caso de uso (véase Sección 1.1). Es importante observar que estas reglas se definen específicamente para este contexto, y cualquier otro conjunto de reglas podría ser definido para adaptarse mejor al tipo de combinaciones y/o transformaciones que sean necesarias en los diferentes casos de uso. La tabla 4.2 presenta cada regla con una descripción y la Figura 4.10 muestra un ejemplo de juguete de aplicación de cada regla.

Nombre	Descripción
Extract	Dada una hipótesis h , esta regla extrae una parte específica de h como conclusión.
Concat	Concatena dos hipótesis independientes para generar la conclusión.
Instantiate	Genera la conclusión mediante el instanciamiento de una hipótesis genérica a un caso particular.
Compose	Combina dos hipótesis que comparten algún elemento en común para generar una nueva conclusión.
Refine	Dada una hipótesis h , adapta levemente su sintaxis para que sirva como respuesta a la pregunta, sin modificar la semántica del contenido de h .
NoInfo	Esta regla es utilizada cuando ninguna de las hipótesis aporta información para responder la pregunta (o parte de ella). No se aplica a ninguna hipótesis.

Tabla 4.2: Lista de reglas definidas con una breve explicación.

(A) Extract	(B) Concat	(C) Instantiate
<p>Hoy jugué al fútbol con unos amigos en la playa y fue muy divertido.</p> <hr/> <p>Hoy jugué al fútbol</p> <p>[Extract]</p>	<p>Deforestación El cambio afecta la climático es un biodiversidad problema global</p> <hr/> <p>La deforestación afecta a la biodiversidad. Además, el cambio climático es un problema global.</p> <p>[Concat]</p>	<p>Los árboles son beneficiosos para el medio ambiente</p> <hr/> <p>Los pinos son beneficiosos para el medio ambiente</p> <p>[Instantiate]</p>
(D) Compose	(E) Refine	(F) NoInfo
<p>Deforestación Biodiversidad es afecta la esencial para la biodiversidad salud del planeta</p> <hr/> <p>La deforestación afecta la salud del planeta.</p> <p>[Compose]</p>	<p><i>Pregunta:</i> Las abejas son importantes en la polinización de las flores, ¿verdad?</p> <hr/> <p>Las abejas desempeñan un papel crucial en la polinización</p> <hr/> <p>Correcto, las abejas desempeñan un papel crucial en la polinización.</p> <p>[Refine]</p>	<p>Se formula una pregunta sin respuesta en los documentos.</p> <hr/> <p>Lo siento, pero no cuento con la información para responder la pregunta.</p> <p>[NoInfo]</p>

Figura 4.10: Ejemplos sencillos de aplicación de cada regla. Los ejemplos (E) y (F) tienen información de la consulta como contexto adicional.

4.3.3. Algoritmo

El Algoritmo 1 presenta el pseudo-código que se debe seguir para construir una derivación. Observar que las líneas 3, 4, 5 y 6 corresponden a los pasos que son responsabilidad del LLM. Las tareas que el modelo de lenguaje debe resolver son:

- Decidir qué regla aplicar (línea 3),
- Decidir qué hipótesis usar (línea 4),
- Construir la conclusión que surge de la aplicación de la regla (línea 5),
- Determinar si la conclusión sirve como respuesta final a la consulta del usuario (condición de la línea 6).

Algoritmo 1 Pseudo-código de prompting con derivaciones

Require: $lista_hipotesis = \{h_1, \dots, h_n\}$: Lista de hipótesis, q : Pregunta

```
1:  $es\_respuesta\_final \leftarrow \mathbf{False}$ 
2: while not  $es\_respuesta\_final$  do
3:   Decidir qué regla  $r$  aplicar
4:   Decidir qué hipótesis  $\{h_i, \dots, h_k\}$  usar en la aplicación de  $r$ 
5:    $conclusion \leftarrow$  Aplicar regla  $r$  con hipótesis  $\{h_i, \dots, h_k\}$  y pregunta  $q$ 
6:   if  $conclusion$  es la respuesta final then
7:      $es\_respuesta\_final \leftarrow \mathbf{True}$ 
8:   else
9:      $lista\_hipotesis.append(conclusion)$ 
10:  end if
11: end while
12: return  $conclusion$ 
```

El resto del algoritmo es directo. Si la conclusión es considerada como la respuesta final, el algoritmo termina y la conclusión es retornada al usuario como respuesta. En caso contrario, se agrega a la lista de hipótesis y puede ser utilizada en siguientes aplicaciones de reglas (aunque potencialmente podría no ser utilizada). Opcionalmente, es posible almacenar cada aplicación de regla con punteros a las hipótesis utilizadas como argumentos así como a su conclusión parcial, para luego poder reconstruir el árbol de derivación.

Se exploraron dos alternativa de implementar el algoritmo 1:

1. **Prompt de un paso:** Esta alternativa aísla cada aplicación de regla como una llamada independiente al LLM. Dada una lista de hipótesis en el medio de la construcción de una derivación, el modelo es utilizado para producir, en un solo paso de inferencia, la regla a aplicar, las hipótesis a usar, la conclusión que surge como resultado, y si dicha conclusión es la respuesta final. Luego, el algoritmo es implementado siguiendo la estructura del pseudo-código, en donde las líneas 3, 4, y 5 son reemplazadas por la llamada al modelo de lenguaje y el parsing de la salida generada.

- Prompt de derivación completa:** En contraste con la otra alternativa, en esta se utiliza al LLM para construir la derivación entera en una sola inferencia, efectivamente emulando la ejecución del Algoritmo 1. Para lograr esto, se aplicó Few-Shot prompting (Brown y cols., 2020), para lo que se construyeron 6 ejemplos de ejecuciones manuales completas del algoritmo para diferentes derivaciones utilizando todas las reglas. A partir de estos ejemplos se diseña una prompt con la instrucción de seguir los mismos pasos de los ejemplos para una nueva consulta e hipótesis iniciales. Luego es necesario hacer parsing del resultado para obtener cada aplicación de regla y conclusiones intermedias.

La prompt construida para ambas alternativas, así como uno de los ejemplos Few-Shot de la segunda alternativa, se encuentran en el Anexo A.4. En los experimentos realizados, la segunda alternativa resultó ser más rápida y barata computacionalmente sin penalizar en la calidad de los resultados. La mejora en eficiencia se debe a que se reemplaza múltiples inferencia del LLM por una sola. A partir de estas observaciones a la hora de realizar los experimentos, se decidió utilizar y evaluar la prompt de derivación completa.

4.3.4. LLMs utilizados

Para este experimento se decidió usar los modelos cerrados Claude 3 Haiku y Claude 3 Opus. Hay dos principales razones por las que no fue posible replicar este experimento con los modelos abiertos:

- **Recursos insuficientes:** Al utilizar Few-Shot prompting con múltiples ejemplos de derivaciones completas, el tamaño de la prompt aumenta considerablemente y no fue posible alcanzar los requerimientos de memoria con los recursos disponibles.
- **Dificultad de la tarea:** Aún probando otros métodos de prompting (como la prompt de un solo paso) para solventar el problema anterior, no fue posible obtener resultados correctos con los modelos abiertos con los que experimentamos (definidos en Sección 4.1.2). La dificultad de la tarea (construir el árbol mediante la aplicación de reglas) parece superar las capacidades de estos modelos.

4.4. Grandes ventanas de contexto

Este experimento consiste en aprovechar el tamaño de las ventanas de contexto de los últimos LLMs que se han puesto a disposición. En el Capítulo 2 del estado del arte se explicó algunos de los métodos existentes para agrandar el tamaño de las ventanas de contexto, sorteando la dificultad que surge debido al aumento de consumo de memoria y cómputo a medida que crece el tamaño del contexto, ya que el mecanismo atencional es de orden cuadrático.

Los grandes modelos de lenguaje de hace dos o tres años (GPT-3, Llama 2, Claude 1) ofrecían un tamaño máximo en torno a los 4k tokens de ventana de

contexto. Esto incluye los tokens ingresados al modelo así como los tokens generados. Sin embargo, a mediados de 2023, Anthropic puso a disposición el modelo **Claude 2** con un tamaño de ventana de contexto de 100k tokens, estableciendo un nuevo estándar². Desde entonces, los modelos que se han ido lanzando han alcanzado e incluso superado esa barrera. Notoriamente, Google ofrece el modelo **Gemini 1.5 Pro**, con un tamaño de ventana de contexto de hasta 2 millones de tokens, lo que equivale a 8 novelas de largo promedio en inglés³.

El objetivo de este experimento es utilizar el modelo de lenguaje generativo directamente, sin realizar un paso de recuperación de información previo. La Figura 4.11 muestra un esquema de este experimento, en donde se observa que todos los documentos son ingresados directamente en la prompt del modelo de lenguaje. Con este experimento se busca responder a la pregunta de si, a medida que se cuenta con LLMs con grandes ventanas de contexto, sigue siendo necesario seguir una estrategia de tipo RAG con una etapa de Retrieval.

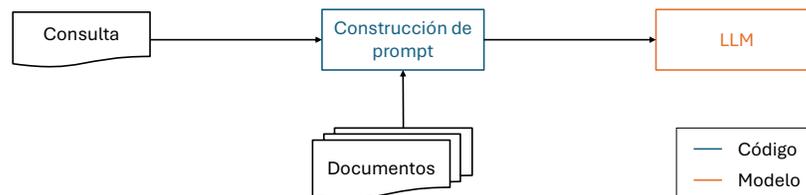


Figura 4.11: Esquema de funcionamiento del experimento de grandes ventanas de contexto.

4.4.1. Construcción de la prompt

El experimento se reduce entonces a construir una prompt muy similar a la utilizada en la etapa de generación de RAG (Sección 4.1.2), que consiste de:

1. Una descripción breve de la tarea a realizar.
2. Todos los documentos completos.
3. Indicaciones de estilo (ser concreto, responder en español, etc).
4. Indicación de responder que no se cuenta con información cuando no es posible responder la pregunta a partir de la información de los documentos.
5. Especificación del formato de la respuesta, que debe incluir una sección al final con las referencias a los documentos utilizados para responder la pregunta.

²<https://www.anthropic.com/news/claude-2>

³<https://ai.google.dev/gemini-api/docs/long-context>

En particular, el punto 2 es la principal diferencia con la prompt utilizada en RAG. Mientras que en RAG se incluyen solamente los *chunks* más similares a la pregunta del usuario, en este experimento no se tiene en cuenta la división por *chunks*, ni se recuperan documentos más similares. En cambio, todos los documentos completos son agregados a la prompt. Se utilizan tags XML `<webpage>` y `</webpage>` para delimitar el inicio y fin de cada documento. Además, al inicio de cada uno se agrega el título del documento y la URL.

La prompt utilizada se encuentra en el Anexo [A.5](#).

4.4.2. LLMs utilizados

La tabla [4.3](#) muestra el tamaño de las ventanas de contexto de cada uno de los modelos utilizados en la etapa de generación de RAG (Sección [4.1.2](#)), así como la cantidad de tokens que resulta de tokenizar todos los documentos para ser agregados a la prompt. Observar que los LLMs que utilizan distintos modelos base (de diferentes proveedores) usan *tokenizers* diferentes, resultando en distintas cantidades de tokens de los documentos tokenizados.

Modelo	Tam. ventana	Tokens de documentos
Mistral 7B Instruct v0.2	32768	22809
OpenChat 3.5 0106	8192	22809
Llama 3 8B Instruct	8192	18369
Claude 3 Haiku	200000	21328
Claude 3 Opus	200000	21328

Tabla 4.3: Comparación por LLM del tamaño de las ventanas de contexto y la cantidad de tokens de los documentos tokenizados.

Analizando los tres grandes modelos de lenguaje abiertos utilizados, se puede observar que el único modelo con un tamaño de ventana de contexto suficientemente grande para cargar los 17 documentos utilizados en nuestro caso de prueba es **Mistral 7b Instruct v0.2**. Aun así, debido a que los modelos abiertos deben ser ejecutados con nuestros recursos, la cantidad de memoria necesaria para aprovechar toda la ventana de contexto del modelo supera la cantidad de memoria disponible para estos experimentos. Debido a esto, solo fue posible realizar este experimento con los modelos cerrados de Anthropic (**Claude 3 Haiku** y **Claude 3 Opus**).

4.4.3. Tiempo de respuesta y costo

Un aspecto importante que se desprende de realizar esta prueba es el impacto en el tiempo de respuesta y el costo de consumo de la API de Anthropic (que sirve como referencia del costo computacional adicional), en comparación con el método RAG.

Método	Modelo	Retrieval (s)	Generación (s)	Costo (US\$)
RAG	Haiku	~1	~1.5	0.0007
Uso de ventana	Haiku	0	~3	0.0058
RAG	Opus	~1	~10	0.0405
Uso de ventana	Opus	0	~20.5	0.3450

Tabla 4.4: Comparación de tiempo y costo de los métodos RAG y uso de ventana de contexto con los modelos Claude 3.

La tabla 4.4 muestra el tiempo aproximado en segundos y el costo en dólares de realizar una consulta utilizando cada uno de los métodos para ambos modelos. Se puede observar que el tamaño del modelo (Opus siendo más grande, aunque se desconocen los tamaños exactos de cada uno) tiene un impacto mayor en el tiempo de respuesta al utilizar una mayor porción de la ventana de contexto. Además, el costo de este método es aproximadamente 8 veces mayor que el método de RAG. Estos dos aspectos pueden ser determinantes a la hora de decidir la solución final a utilizar en un sistema real.

El análisis de resultados al evaluar el método y su comparación con RAG se encuentra en el Capítulo 5.

4.5. Síntesis

Es este capítulo se presentó una descripción detallada de los experimentos realizados para cada uno de los métodos propuestos. En el capítulo siguiente se define la metodología de evaluación, se presentan los resultados obtenidos y se realiza un análisis detallado para cada uno de los métodos explorados.

Capítulo 5

Evaluación y Resultados

En este capítulo se presenta la metodología de evaluación llevada a cabo y los resultados obtenidos para cada uno de los experimentos realizados. La Sección 5.1 se centra en el dataset de evaluación y las métricas utilizadas para evaluar los experimentos, y la Sección 5.2 presenta los resultados obtenidos para cada uno de los experimentos, así como un análisis detallado de los resultados de cada método explorado. Por último, la Sección 5.3 se centra en la evaluación humana realizada para respaldar los resultados obtenidos con las métricas automáticas.

5.1. Evaluación

En esta sección se explica la metodología de evaluación automática que se llevó a cabo de la misma manera para todos los experimentos realizados. También se realizó evaluación humana, la cual se reporta en la Sección 5.3.

En primer lugar, en la Subsección 5.1.1 se detalla el proceso de obtención y curación del dataset de evaluación sobre el cual se evaluaron los experimentos. Luego, en la Subsección 5.1.2 se describen los métodos y métricas de evaluación que se utilizaron para evaluar.

5.1.1. Dataset de evaluación

Para realizar una evaluación sobre si el contenido de las respuestas generadas es correcto, es necesario contar con un dataset de preguntas y respuestas reales, que hayan sido realizadas por estudiantes de la Facultad y respondidas por quienes están a cargo de atender este tipo de consultas. Para ello se recurrió al Espacio de Orientación y Consulta (EOC), quienes offician de clientes en el proyecto. Como se explicó en la Sección 1.1, el EOC es la unidad de la Facultad encargada de atender las consultas de los estudiantes y brindar información sobre la Facultad, entre otras actividades.

Hoy en día, el EOC pone a disposición una dirección de correo electrónico a la cual los estudiantes (o cualquier persona interesada) pueden enviar un mail

para realizar una consulta, y obtener una respuesta personalizada. Se decidió entonces, con el permiso del EOC, aprovechar el conjunto de mails enviados en el último tiempo para generar un dataset de preguntas y respuestas reales que se parecerán mucho a las que se busca responder en el sistema QA.

Se recabaron 167 mails enviados entre agosto y diciembre de 2023. Se pretende generar un conjunto de pares ⟨Pregunta, Respuesta⟩ a partir de estos mails. Sin embargo, estos presentan ciertos desafíos, principalmente:

1. Muchos de ellos tienen más de dos interacciones.
2. Poseen información propia de un mail que no es muy apropiada para el tipo de sistema que se pretende diseñar (por ejemplo: saludos, disculpas por la molestia, etc).
3. Algunos de los mails contienen información personal (cédula, teléfono, etc) que sería conveniente quitar.

La Figura 5.1 muestra la distribución de los mails según la cantidad de interacciones de cada uno de ellos, y se puede observar que 135 mails contienen dos o tres interacciones, mientras que 32 poseen más de tres. Analizando los mails, se observa lo siguiente:

- **Mails de dos interacciones:** Son los que mejor se adaptan al formato ⟨Pregunta, Respuesta⟩.
- **Mails de tres interacciones:** También se adaptan correctamente a este formato, ya que el tercer mail suele ser enviado por la persona que realizó la consulta, agradeciendo por la respuesta.
- **Mails de más de tres interacciones:** Suelen seguir un formato más parecido a una conversación, y por ende resulta más complicado extraer un par ⟨Pregunta, Respuesta⟩.

En función de esto, se decidió acotar el conjunto de mails a utilizar a los que consisten en dos o tres interacciones, y se descarta el tercer mail en el caso de tres interacciones. Esto resuelve el primer desafío mencionado previamente.

Por otro lado, para mitigar los desafíos (2) y (3) mencionados, se decidió realizar una limpieza de los 135 mails restantes. Para ello, se utilizó el LLM `Llama-2-7b-chat-hf`¹. Se diseñó una prompt para procesar los mails que contienen las consultas, en la que se instruye al modelo a identificar y extraer la pregunta a partir del mail. Se indica que se debe eliminar cualquier contenido del correo que no sea la pregunta en sí, pero se especifica que se debe alterar lo menos posible la formulación de la pregunta original. Además, se agregan dos ejemplos de la tarea resuelta (Few-Shot learning). La prompt utilizada se puede consultar en el Anexo A.6. De esta manera, se elimina toda información personal de los mails y se transforman a un formato de consulta más adecuado para un sistema QA.

¹<https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>



Figura 5.1: Gráfica de barras que contiene la distribución de los mails en función de la cantidad de interacciones.

Para los mails que son respuestas a las consultas, no se realizó el mismo procesamiento. En cambio, se definieron una serie de heurísticas a partir de analizar varios ejemplos de los mails y notar cierta regularidad en el formato. En particular, todos ellos comienzan igual (“Hola «nombre»,”) y terminan igual (“Saludos, «nombre»”), por lo que se decidió eliminar esas partes de los mails utilizando expresiones regulares.

Luego de este procesamiento, se obtuvo un conjunto de 135 instancias de pares ⟨Pregunta, Respuesta⟩ que componen el dataset de evaluación utilizado para evaluar todos los experimentos realizados.

5.1.2. Métricas de evaluación

Una vez obtenido el dataset de evaluación, es necesario definir las métricas con las que se va a evaluar las respuestas generadas con los distintos experimentos. A partir de lo desarrollado en el Capítulo 2 del Estado del Arte, se decidió explorar tres familias de métricas diferentes:

1. **Métricas basadas en reglas:** BLUE, Rouge-1, Rouge-2 y Rouge-L. Estas métricas se basan en la comparación de coincidencias de n-gramas entre el texto generado y el texto de referencia.
2. **Métricas basadas en BERT:** BERTScore (T. Zhang, Kishore, Wu, Weinberger, y Artzi, 2020), del que se obtienen valores de Precision, Recall y F1 Score. Estas métricas utilizan la representación vectorial contextualizada de las palabras obtenida por BERT para comparar la similitud semántica entre el texto generado y el texto de referencia.
3. **Métricas utilizando un LLM como evaluador:** Modelo de lenguaje cerrado Claude 3 Opus. Se utiliza el modelo para puntuar la respuesta generada en función de la respuesta de referencia. Se explicará en detalle más adelante en esta sección.

Probando estas métricas se observó que el comportamiento de las primeras dos familias de métricas muchas veces no era el esperado, y no se correspondía con el criterio humano. Para mostrar este fenómeno, vamos a dar un ejemplo en donde esto queda en evidencia.

Supongamos que tenemos la siguiente **pregunta** (extraída del dataset de evaluación): “*Es posible estudiar dos carreras a la par, como ing en sistemas e ing química?*”

Cuya **respuesta correcta** es: “*No hay problema. Puedes abrir dos carreras, y las materias que sean comunes a ambas se revalidan de una carrera para la otra de forma automática.*”

Ahora, supongamos que tenemos un **Sistema 1** que genera la siguiente **respuesta candidata**: “*Sí, es posible estudiar Ingeniería en Sistemas e Ingeniería Química en simultáneo. Si lo haces, ten en cuenta que las unidades curriculares (materias) que apruebes y sean comunes a ambas carreras se validarán automáticamente en las dos escolaridades.*”

Además, supongamos que tenemos un **Sistema 2** que genera esta otra **respuesta candidata**: “*No, no es posible estudiar Ingeniería en Sistemas e Ingeniería Química en simultáneo. Ten en cuenta que las unidades curriculares (materias) que apruebes y sean comunes a ambas carreras solo se validarán para la escolaridad correspondiente a la que estés cursando.*”

Analizando la respuesta correcta, hay dos elementos de información fundamentales que deberían aparecer en las respuestas candidatas para ser calificadas como correctas, los cuales se encuentran subrayados, y son: (1) sí es posible estudiar las dos carreras en simultáneo, (2) las materias comunes a ambas carreras se revalidan.

Mirando las dos respuestas candidatas, es claro que la generada por el Sistema 1 correctamente incluye los dos elementos, y la generada por el Sistema 2 incluye dos elementos de información que son incorrectos, ya que son información contradictoria a la esperada. Sin embargo, es importante notar que ambas respuestas están redactadas de forma casi idéntica, utilizando muchas palabras en común. La diferencia es mayoritariamente semántica, por lo que para dar un resultado que se corresponda con el criterio humano, las métricas deberían ser capaces de capturar la semántica de las respuestas. Entonces, sería de esperar que la respuesta generada por el Sistema 1 tuviera un resultado cercano a 1 y la respuesta generada por el Sistema 2 tuviera un resultado cercano a 0.

La Tabla 5.1 muestra que los resultados obtenidos para este ejemplo con las métricas basadas en reglas y BERTScore presentan varias deficiencias, que se explican a continuación.

Sistema	BLEU	Rouge-1	Rouge-2	Rouge-L	BERTScore F1
Sistema 1	0.071	0.338	0.116	0.254	0.714
Sistema 2	0.076	0.351	0.139	0.297	0.700

Tabla 5.1: Resultados obtenidos con las métricas basadas en reglas y BERTScore para las respuestas candidatas de los sistemas del ejemplo.

Por un lado, las métricas basadas en reglas evalúan más alto a la respuesta incorrecta (Sistema 2). Además, los valores son bastante bajos para ambas respuestas, especialmente en el caso de BLEU. Este resultado no debería sorprendernos demasiado, ya que estas métricas se basan en contar frecuencias de N-gramas repetidos entre la respuesta candidata y la respuesta de referencia. Debido a que las respuestas generadas son bastante diferentes a la respuesta correcta en cuanto a la forma en que están redactadas, es de esperar que los valores sean bajos.

Además, notar que la respuesta correcta utiliza la frase “No hay problema”, que contiene una negación, para afirmar que sí es posible anotarse a ambas carreras. La respuesta generada por el Sistema 2 también tiene una negación en “No, no es posible”, pero su semántica es exactamente la contraria. Aun así, es probable que esto influya positivamente en el resultado final de estas métricas por la coincidencia de la negación. Por otro lado, la respuesta generada por el Sistema 1 no tiene dicha negación, lo que le podría penalizar en su puntuación final.

Analizando los resultados obtenidos con BERTScore, si bien el Sistema 1 obtiene correctamente el valor más alto, aun así los dos resultados son muy similares, con una diferencia de 0.014. Además, ambos valores son relativamente altos, cuando se esperaría que el valor para la respuesta generada por el Sistema 2 fuera cercano a 0.

En general, en estas dos familias de métricas, se observó que los resultados varían muy poco entre sí para los diferentes experimentos (como se puede notar en la Tabla 5.4 de la Sección 5.2). Esto complejiza el análisis ya que es muy difícil interpretar los resultados obtenidos.

Es por esto que resulta muy importante buscar una métrica un poco más fiable e interpretable, y el uso de LLMs para la tarea de evaluación es una posible alternativa. Lo que resta de esta sección consiste en explicar el diseño de esta métrica, y una breve comparación con las otras métricas utilizando el mismo ejemplo ya definido.

Evaluación con LLM

Se ha mostrado que los LLMs más capaces tienden a exhibir un alto grado de acuerdo con la evaluación humana cuando son utilizados como evaluadores (LLM-as-a-Judge) (Zheng y cols., 2023). A partir de esta idea, se diseñó una prompt de evaluación siguiendo el formato definido para el *Feedback Collection Dataset* (Kim y cols., 2024), un dataset diseñado para hacer fine-tuning de LLMs con el fin de mejorar la capacidad de evaluación de los modelos más chicos. El formato que se siguió consiste de cuatro componentes:

1. **Instrucción a evaluar:** La instrucción de la tarea que será evaluada. En nuestro caso, esta es la pregunta particular para la que se generó la respuesta a evaluar.
2. **Respuesta a evaluar:** La respuesta candidata a la pregunta que el LLM debe evaluar con un score en la escala del 1 al 5.

Score	Explicación	Clasificación
1	Candidata contradice referencia, información falsa .	Inaceptable
2	Candidata tiene conflictos con referencia, información parcialmente falsa .	Inaceptable
3	Candidata no contradice referencia pero tampoco aporta ninguna información.	Aceptable
4	Candidata coincide parcialmente con referencia, información correcta pero incompleta .	Aceptable
5	Candidata coincide completamente con referencia, información correcta y completa .	Aceptable

Tabla 5.2: Breve explicación de cada uno de los scores posibles, definidos en la rúbrica de puntuación propuesta en este trabajo.

3. **Respuesta de referencia:** La respuesta de referencia (respuesta correcta), a la que le correspondería un score de 5.
4. **Rúbrica de puntuación específica:** Criterio específicamente definido para nuestro caso de uso, especificando los aspectos que el LLM debería tener en cuenta. Esto incluye una descripción del criterio, y una explicación detallada sobre cada uno de los scores posibles (1 al 5).

El criterio de la rúbrica diseñada consiste en determinar si la respuesta generada es correcta y veraz. Esto se especifica claramente en la descripción de cada score. La tabla 5.2 presenta una breve explicación de cada puntuación. Adicionalmente, se utiliza el método de Chain-of-Thought (CoT) (Wei y cols., 2024) en formato Zero-Shot (sin ejemplos, solo indicado mediante instrucciones), de forma que el modelo debe generar una justificación de su evaluación antes de asignar un score. La prompt utilizada se puede leer en el Anexo A.7.

Adicionalmente, se definió una clasificación binaria que consiste en agrupar los scores 1 y 2 como respuestas inaceptables, y los scores 3 a 5 como respuestas aceptables. Los scores 1 y 2 corresponden a las respuestas que contradicen total o parcialmente la respuesta de referencia, dando información incorrecta, y por lo tanto son consideradas como inaceptables. El score 3, en cambio, corresponde al caso particular en que la respuesta no aporta nada de información, ni correcta ni incorrecta. Aquí deberían ser clasificadas las respuestas que indican que no se cuenta con información, respuestas que no son útiles pero tampoco dañinas. Los scores 4 y 5 corresponden a respuestas útiles, que aportan parte o toda la información correcta. En estos tres casos (scores 3 a 5), las respuestas no son incorrectas ni tienen información contradictoria, por lo que son consideradas como aceptables.

Esta agregación nos permite tener una forma muy directa de interpretar la métrica, y nos da un objetivo concreto al que apuntar: buscaremos maximizar la cantidad de respuestas aceptables.

Veamos como se comporta esta métrica con el ejemplo presentado anteriormente en esta sección.

En primer lugar, observemos el razonamiento (CoT) generado para la respuesta candidata del Sistema 1: “*La respuesta a evaluar coincide completamente con la respuesta de referencia de una manera veraz y tiene toda la información que se expresa en la respuesta de referencia. Ambas respuestas afirman que es posible estudiar dos carreras simultáneamente y que las materias comunes se revalidan automáticamente entre las dos carreras.*”. Finalmente, el modelo le asigna el **score 5**.

Veamos ahora el razonamiento (CoT) generado para la respuesta candidata del Sistema 2: “*La respuesta a evaluar contradice la respuesta de referencia de una manera que la información no es veraz en comparación con la respuesta de referencia. La respuesta a evaluar indica que no es posible estudiar Ingeniería en Sistemas e Ingeniería Química simultáneamente, mientras que la respuesta de referencia afirma que sí es posible abrir dos carreras y que las materias comunes se revalidan automáticamente de una carrera a otra.*”. A esta respuesta, el modelo le asigna el **score 1**.

Se puede observar que los razonamientos generados coinciden en gran medida con el análisis realizado previamente sobre estas dos respuestas, y los scores asignados se alinean perfectamente con lo esperado. Si bien esto es solo un ejemplo, sirve como ilustración de las ventajas, tanto en el resultado obtenido como en la interpretabilidad del mismo, de utilizar al LLM como evaluador por sobre las otras métricas exploradas.

5.2. Resultados

A continuación se reportan los resultados de 25 experimentos distintos que se enmarcan dentro de los cuatro métodos previamente explicados en el Capítulo 4 de experimentación.

La Tabla 5.3 corresponde a la evaluación realizada utilizando Claude 3 Opus como modelo evaluador, según fue explicado en la Sección 5.1. Se muestra la cantidad de respuestas para los distintos scores, el promedio de score, el promedio normalizado en el rango $[0, 1]$ y la desviación estándar. La última columna (Aceptables) corresponde a la proporción de respuestas consideradas aceptables (con score de 3, 4, o 5). Por otro lado, la Tabla 5.4 presenta las métricas Rouge (1, 2 y L), BLEU, y BERTScore (Precision, Recall y F1 Score).

En ambas Tablas, **BE** corresponde al uso de *Bi-Encoder* y **CE** al uso de *Cross-Encoder* como modelo de Retrieval, **k** es la cantidad de *chunks* recuperados en cada consulta y se muestran resaltados los mejores valores obtenidos en cada métrica.

Haciendo una lectura general de los resultados, se desprenden algunas primeras conclusiones:

- Los mejores resultados de todas las métricas se dividen entre los experimentos de Derivaciones y RAG. En particular, los dos mejores experimentos fueron usando Claude Opus (el modelo más capaz) y $k = 7$.

Método	k	LLM	#1	#2	#3	#4	#5	Avg	N avg	Stdev	Aceptables
RAG BE	3	OpenChat 3.5	47	20	51	14	3	2.3	0.33	1.12	0.504
RAG CE	3	OpenChat 3.5	44	19	53	17	2	2.36	0.34	1.11	0.533
RAG BE	3	Llama 3 8B	40	19	64	10	2	2.37	0.34	1.03	0.563
RAG CE	3	Llama 3 8B	35	22	55	21	2	2.5	0.38	1.08	0.578
RAG BE	3	Mistral 7B	33	24	62	16	0	2.45	0.36	0.99	0.578
RAG CE	3	Mistral 7B	28	20	54	30	3	2.7	0.43	1.1	0.644
RAG CE	5	Mistral 7B	34	22	55	23	1	2.52	0.38	1.07	0.585
RAG CE	7	Mistral 7B	33	18	59	22	3	2.59	0.4	1.1	0.622
RAG BE	3	Claude Haiku	19	17	80	16	3	2.76	0.44	0.92	0.733
RAG CE	3	Claude Haiku	17	20	74	20	4	2.81	0.45	0.94	0.726
RAG CE	5	Claude Haiku	14	18	76	23	4	2.89	0.47	0.91	0.763
RAG CE	7	Claude Haiku	15	17	71	24	8	2.95	0.49	0.99	0.763
RAG CE	9	Claude Haiku	15	15	70	29	6	2.97	0.49	0.98	0.778
RAG BE	3	Claude Opus	15	15	80	22	3	2.87	0.47	0.89	0.778
RAG CE	3	Claude Opus	15	15	75	27	3	2.91	0.48	0.92	0.778
RAG CE	7	Claude Opus	15	8	71	32	9	3.09	0.52	1.0	0.83
Fine-tuning	3	Llama 3 8B	55	14	51	10	5	2.23	0.31	1.17	0.489
Fine-tuning	3	Mistral 7B	36	25	56	15	3	2.44	0.36	1.07	0.548
Derivación BE	3	Claude Haiku	21	3	95	12	4	2.81	0.45	0.91	0.822
Derivación CE	3	Claude Haiku	20	16	83	13	3	2.73	0.43	0.91	0.733
Derivación BE	3	Claude Opus	15	8	93	15	4	2.89	0.47	0.85	0.83
Derivación CE	3	Claude Opus	10	4	92	25	4	3.07	0.52	0.79	0.896
Derivación CE	7	Claude Opus	9	5	93	22	6	3.08	0.52	0.8	0.896
Uso de ventana	-	Claude Haiku	35	12	61	22	5	2.63	0.41	1.14	0.652
Uso de ventana	-	Claude Opus	17	15	73	25	5	2.9	0.47	0.97	0.763

Tabla 5.3: Resultados de cada experimento evaluando con Claude Opus.

Método	k	LLM	Rouge1	Rouge2	RougeL	BLEU	P	R	F1
RAG BE	3	OpenChat 3.5	0.247	0.058	0.153	0.068	0.665	0.676	0.669
RAG CE	3	OpenChat 3.5	0.247	0.059	0.154	0.058	0.668	0.677	0.671
RAG BE	3	Llama 3 8B	0.237	0.05	0.145	0.041	0.656	0.672	0.663
RAG CE	3	Llama 3 8B	0.247	0.054	0.15	0.043	0.656	0.678	0.666
RAG BE	3	Mistral 7B	0.241	0.054	0.148	0.053	0.666	0.672	0.668
RAG CE	3	Mistral 7B	0.243	0.053	0.148	0.045	0.657	0.675	0.664
RAG CE	5	Mistral 7B	0.244	0.056	0.15	0.044	0.656	0.677	0.665
RAG CE	7	Mistral 7B	0.248	0.058	0.154	0.049	0.658	0.679	0.667
RAG BE	3	Claude Haiku	0.24	0.058	0.145	0.039	0.654	0.681	0.667
RAG CE	3	Claude Haiku	0.235	0.051	0.139	0.032	0.654	0.682	0.667
RAG CE	5	Claude Haiku	0.237	0.054	0.144	0.038	0.653	0.682	0.667
RAG CE	7	Claude Haiku	0.241	0.059	0.145	0.042	0.655	0.686	0.669
RAG CE	9	Claude Haiku	0.245	0.06	0.147	0.04	0.656	0.689	0.672
RAG BE	3	Claude Opus	0.254	0.054	0.149	0.042	0.661	0.683	0.671
RAG CE	3	Claude Opus	0.254	0.059	0.151	0.043	0.66	0.686	0.672
RAG CE	7	Claude Opus	0.26	0.066	0.155	0.044	0.662	0.695	0.677
Fine-tuning	3	Llama 3 8B	0.241	0.055	0.143	0.041	0.65	0.685	0.666
Fine-tuning	3	Mistral 7B	0.234	0.053	0.138	0.039	0.645	0.686	0.664
Derivación BE	3	Claude Haiku	0.268	0.059	0.161	0.062	0.67	0.685	0.676
Derivación CE	3	Claude Haiku	0.261	0.056	0.155	0.055	0.668	0.68	0.673
Derivación BE	3	Claude Opus	0.251	0.058	0.153	0.069	0.67	0.676	0.672
Derivación CE	3	Claude Opus	0.263	0.067	0.157	0.062	0.668	0.684	0.674
Derivación CE	7	Claude Opus	0.269	0.067	0.157	0.063	0.663	0.683	0.671
Uso de ventana	-	Claude Haiku	0.255	0.062	0.151	0.046	0.66	0.686	0.672
Uso de ventana	-	Claude Opus	0.257	0.06	0.151	0.05	0.656	0.683	0.669

Tabla 5.4: Resultados de cada experimento con las métricas Rouge, BLEU y BERTscore.

- El método de Derivaciones es notoriamente el que obtiene una mayor proporción de respuestas aceptables. El mejor resultado de Derivaciones en esta métrica es de **0.896**, frente a **0.83** como mejor resultado del resto. Esto también se puede corroborar mirando la métrica Precision, en donde este método también obtiene los mejores resultados.
- El método de RAG usando Claude Opus y $k = 7$ obtiene mayor cantidad de respuestas perfectas (score 5), y mayor Recall (y F1 Score).
- Como era de esperarse, los LLMs cerrados obtienen significativamente mejores resultados que los modelos abiertos probados en estos experimentos. Además, el orden de los resultados se corresponde con las capacidades esperadas de los modelos (Claude Opus obtiene los mejores resultados siendo el modelo más capaz).
- Como era de esperarse, en los experimentos que usan Retrieval, comparando el mismo método y LLM, utilizar modelos de tipo *Cross-Encoder* suele dar mejores resultados que utilizar modelos tipo *Bi-Encoder*.
- El aumento de la cantidad de *chunks* recuperados (k) se corresponde con mejores resultados en los modelos Claude Haiku y Claude Opus, pero dicha mejora no es apreciable en Mistral 7B.

En las siguientes secciones de este Capítulo se abordan distintos análisis realizados en función de cada método explorado. Nos basaremos principalmente en la evaluación realizada utilizando Claude Opus, ya que es la más sencilla de interpretar, y además es la que ofrece resultados más consistentes (como ya se explicó en la Sección 5.1).

5.2.1. Retrieval-Augmented Generation

A continuación se presentan algunas observaciones de los resultados obtenidos con los experimentos que utilizan el método RAG.

En primer lugar, como ya se comentó antes, se observa que en todos los experimentos, bajo el mismo LLM y valor de k , los resultados utilizando un modelo *Cross-Encoder* en el paso de Retrieval son siempre superiores a los obtenidos usando un modelo *Bi-Encoder* (mirando el promedio de los scores). Esto es consistente con lo explicado en el Estado del Arte (Capítulo 2). Además, es una señal de que el modelo *Cross-Encoder* está recuperando *chunks* más relevantes en ciertos casos, lo que beneficia a todos los modelos (y por ende lleva a una mejora en todos los casos).

Nos centraremos ahora en los experimentos utilizando el modelo *Cross-Encoder*, fijando $k = 3$ (valor con el que se realizaron la mayoría de los experimentos). La Figura 5.2 presenta gráficas de frecuencia de los scores y cantidad de aceptables e inaceptables para los distintos modelos, con dichos parámetros.

Una primera observación evidente es que hay una mejora en la proporción de respuestas aceptables a medida que se utiliza un modelo más capaz (considerando los *benchmarks* publicados para cada modelo), a excepción del caso de

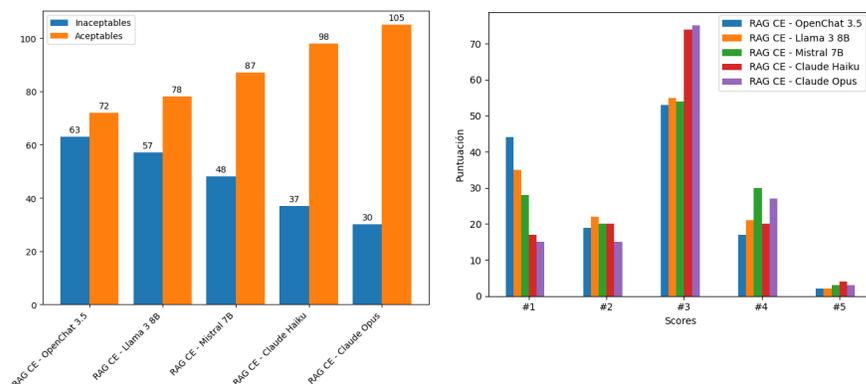


Figura 5.2: Gráficas correspondientes a experimentos usando la técnica RAG con *Cross-Encoder* y $k = 3$. Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

Llama 3 8B que presenta peores resultados que Mistral 7B. Luego, los modelos de la familia Claude 3 (modelos cerrados) son considerablemente superiores a los modelos abiertos utilizados en estos experimentos.

Sin embargo, observando en detalle la gráfica de la derecha, se puede ver que la mayor diferencia entre los distintos modelos se encuentra en el score 3 (que corresponde a respuestas que no aportan información pero tampoco dan información falsa, respuestas del tipo “No tengo información para responder esa pregunta”). Los modelos Claude tienen en torno a 20 respuestas más en dicha categoría respecto a los modelos abiertos. Además, esa considerable diferencia también se puede observar en el Score 1 (un poco en menor medida). No obstante, no hay una diferencia considerable en las respuestas con scores 4 y 5. Más aún, se observa que Mistral 7B supera en cantidad de respuestas en estas categorías (33) a Claude Haiku (24) e incluso a Claude Opus (30). Aun así, la penalización que sufre Mistral 7B en la generación de respuestas inaceptables (scores 1 y 2) es muy grande (48 frente a 37 de Haiku y 30 de Opus).

Esto se puede interpretar como que los modelos Claude están realizando menos alucinaciones o razonamientos incorrectos que podrían llevar a respuestas erróneas. Más aún, podría tratarse de preguntas para las cuales no existe información en los documentos para generar una respuesta correcta. En tal caso, los modelos Claude estarían mostrando una mejor capacidad para detectar estos casos y responder que no cuentan con dicha información, mientras que el resto de modelos estaría cayendo en responder algo, aunque sea incorrecto.

Para validar esta hipótesis, se hizo una anotación manual de 53 preguntas de las 135 que componen el dataset de evaluación, clasificándolas en si cuenta con información en los documentos para ser respondidas o no. Esta no es una tarea sencilla ya que es necesario buscar en todos los documentos la posible respuesta a la pregunta. Se utilizó una escala del 1 al 5, donde 1 corresponde con que no hay nada de información en los documentos que ayude a responder

la pregunta, y 5 es que toda la información de la respuesta de referencia puede ser encontrada en los documentos.

La Figura 5.3 presenta en una gráfica de barras la cantidad de preguntas que quedaron en cada categoría. Como se puede observar, la categoría con mayor cantidad de preguntas es efectivamente la que corresponde a preguntas sin información (1). Esto añade un desafío adicional a nuestro problema y hace aún más relevante la distinción entre respuestas inaceptables y aceptables, ya que en muchos casos la mejor respuesta posible dados los documentos disponibles se corresponde con el score 3.

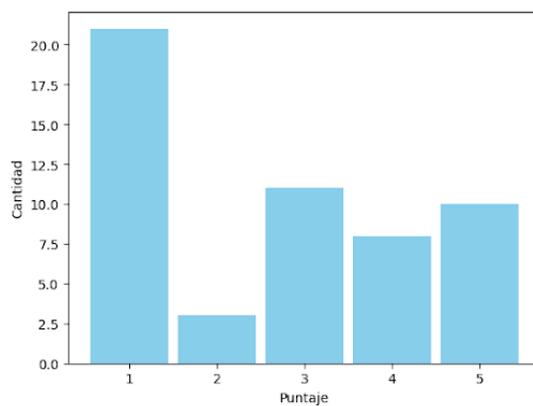


Figura 5.3: Distribución de las preguntas en la escala 1 a 5 según la anotación realizada para determinar la información disponible en los documentos.

Haciendo un análisis cualitativo de las respuestas inaceptables, se observó que efectivamente hay una cierta correlación entre estos scores bajos y la falta de información para responder sus correspondientes preguntas. En otras palabras, la principal razón por la que los modelos generan respuestas inaceptables es la falta de información en los documentos para responder a las respectivas preguntas, lo que lleva a alucinaciones, o un mal uso de la información no relevante de los *chunks* en el contexto (razonamientos erróneos).

Análisis de los distintos valores de k

Se realizaron algunos experimentos para determinar el comportamiento del método RAG, variando la cantidad de *chunks* recuperados utilizando el modelo *Cross-Encoder* (valor k).

Es interesante observar que el comportamiento de los distintos LLMs no es consistente.

La Figura 5.4 muestra las gráficas de frecuencia de scores y proporción de respuestas aceptables/inaceptables para *Mistral 7B* con $k \in \{3, 5, 7\}$. Resulta sorprendente que el aumento de k lleva a una disminución de respuestas con scores 4 y 5, y un aumento del score 1. Una posible explicación para este

fenómeno es que introducir más *chunks* al contexto, aunque pueda en algunos casos traer más información relevante, en muchos otros implicará la inclusión de información no relevante a la pregunta en el contexto. Es entonces posible que en muchas de estas preguntas, la información no relevante esté llevando al modelo a producir respuestas con información incorrecta.

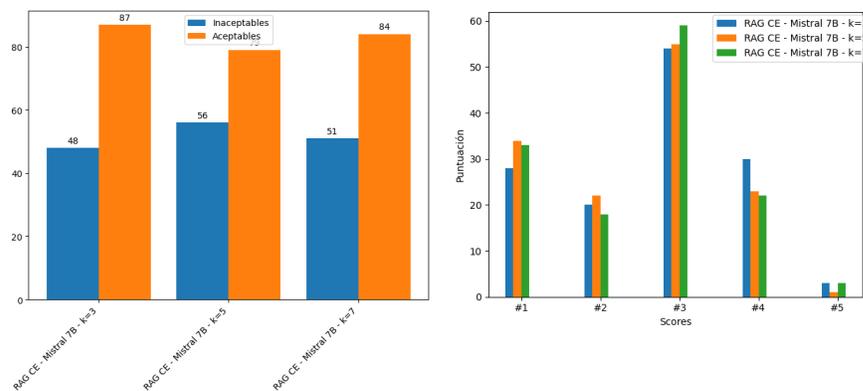


Figura 5.4: Gráficas correspondientes a experimentos usando la técnica RAG con el modelo *Mistral 7B* con *Cross-Encoder*, para distintos valores de k . Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

Por otro lado, la Figura 5.5 muestra las mismas gráficas pero para los modelos *Claude Haiku* con $k \in \{3, 5, 7, 9\}$, y *Claude Opus* con $k \in \{3, 7\}$. Para estos modelos el comportamiento es bien distinto. A medida que el valor de k aumenta, la cantidad de respuestas inaceptables disminuye. Además, las respuestas con scores 4 y 5 aumentan. Notoriamente, *Claude Haiku* con $k = 9$ termina sobrepasando los resultados de *Claude Opus* con $k = 3$ (promedio de scores de 2.97 frente a 2.91, y misma proporción de respuestas aceptables). Por otro lado, *Claude Opus* con $k = 7$ es el experimento que obtiene mayor cantidad de respuestas con score 5 (9) de todos los experimentos realizados, considerando todos los métodos.

Una conclusión posible respecto a la disparidad del comportamiento de los modelos *Claude* frente a *Mistral 7B* al variar k , es que los primeros son mejores ignorando la información no relevante, y solo se ven beneficiados de la información adicional en los casos en que el aumento de k lleva a recuperar más información relevante (que con un valor de k más chico se perdería).

5.2.2. Fine-tuning

En cuanto a los resultados obtenidos para los experimentos que consistieron en realizar fine-tuning de LLMs a partir del dataset sintético, la primera observación es que este proceso resulto perjudicial comparando con realizar RAG con los modelos sin fine-tuning. La Figura 5.6 muestra las gráficas de frecuencia de

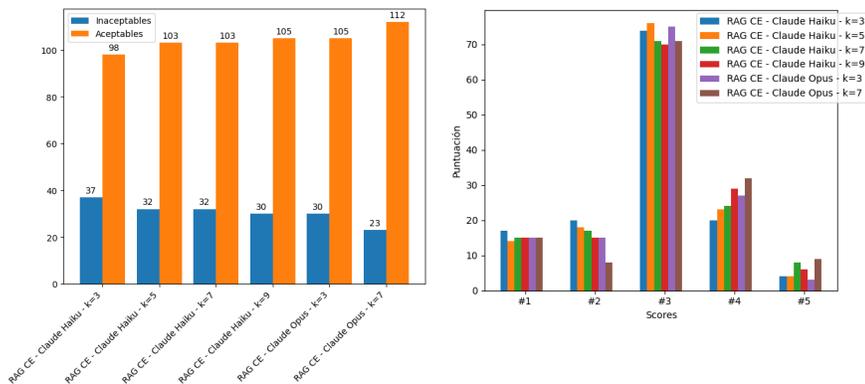


Figura 5.5: Gráficas correspondientes a experimentos usando la técnica RAG con los modelos **Claude Haiku** y **Claude Opus** con *Cross-Encoder*, para distintos valores de k . Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

scores y proporción de respuestas aceptables/inaceptables para **Mistral 7B** sin fine-tuning (corresponde al experimento de RAG con *Cross-Encoder*) así como la versión con fine-tuning, y lo mismo para el modelo **Llama 3 8B**.

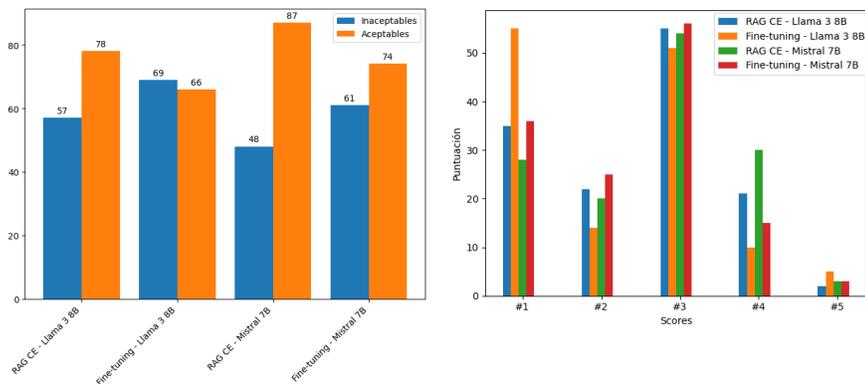


Figura 5.6: Gráficas correspondientes a experimentos usando los métodos RAG y Fine-tuning con los modelos **Llama 3 8B** y **Mistral 7B** con *Cross-Encoder*. Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

En ambos modelos se observa un deterioro en las respuestas luego de realizar el fine-tuning. Hay una disminución de respuestas aceptables, un claro incremento de respuestas con score 1, y una disminución de respuestas con score 4.

Haciendo un análisis cualitativo de las respuestas generadas, se observaron algunos comportamientos que podrían explicar este deterioro.

En primer lugar, parece haber un aumento en las alucinaciones. La figura 5.7

muestra un ejemplo de pregunta en donde la respuesta generada con la versión con fine-tuning de **Mistral 7B** inventa dos unidades o departamentos (marcados en rojo). En cambio, la respuesta utilizando RAG sin fine-tuning es más acotada y específica correctamente que no se cuenta con la información para determinar el sector al que se deberían dirigir (marcado en verde).

Pregunta
Para incorporar un colaborador en modalidad de pasantía educativa de acuerdo a la ley de empleo juvenil, ¿Debemos contactar con la universidad para completar distintos documentos? ¿Es el sector correspondiente o dónde debemos dirigirnos?

Respuesta (Fine-tuning)
Para iniciar un trámite de incorporación de un colaborador en modalidad de pasantía juvenil, debes contactar primero con la **Empresa Pública de Formación (EPF)** que se encarga de la administración de los convenios y las pasantías.

Ellos te guiarán en la elaboración de los distintos documentos necesarios para iniciar el trámite, como el contrato de pasantía, el plan de trabajo, entre otros.

También te recomiendan contactar con la **Unidad de Planes de Estudios (UPE)** de la Facultad de Ingeniería para confirmar los requisitos curriculares y el calendario de inscripción a las materias correspondientes.

Respuesta (RAG)
Sí, para incorporar a un colaborador en modalidad de pasantía educativa en la Facultad de Ingeniería de la Universidad de la República de Uruguay, es necesario contactar con la universidad para completar los documentos correspondientes. **No se especifica en los extractos a qué sector o departamento dirigirse en particular.**

Figura 5.7: Ejemplo de pregunta con las respuestas generadas por la versión con fine-tuning y sin fine-tuning de **Mistral 7B**. Se presenta al inicio parte de un *chunk* de contexto.

Por otro lado, en contraposición con lo que se buscaba al realizar este experimento, parece haber un deterioro en las capacidades de los modelos de utilizar correctamente el contexto. Hay una mayor tendencia a utilizar toda la información que se dispone, aplique o no a la pregunta en cuestión. La Figura 5.8 presenta un ejemplo en donde el modelo con fine-tuning de **Llama 3 8B** utiliza información provista en un *chunk* del contexto, que no es relevante para la pregunta. Incorrectamente, se indica que el trámite de cambio de parcial se inicia en la Mesa de Entrada (marcado en rojo). En cambio, si bien la respuesta del experimento RAG agrega información no relevante, no la utiliza de forma incorrecta, y claramente se especifica que no se cuenta con información para responder la pregunta (marcado en verde). Notar que no se hace mención a Mesa de Entrada en esta respuesta.

Una posible explicación para estos fenómenos observados en los modelos luego de realizar fine-tuning, es que estos adquieren una falsa seguridad en el dominio particular de las preguntas. Contrario a lo esperado, esta seguridad lleva a que utilicen peor los *chunks* del contexto y generen más alucinaciones, ya que aumenta su conocimiento a priori del dominio, aunque de forma defectuosa y deficiente. Los modelos sin fine-tuning, al no haber sido entrenados con un conjunto de preguntas y respuestas muy similares a las que se pretende que respondan, se ven menos sesgados en sus respuestas, lo que les permite tener mayor objetividad en el proceso de resolución de la tarea (responder preguntas

Contexto
 ...
 MESA DE ENTRADA en el caso de: Ingreso - Inscripción extracurricular - Adecuación curricular - Reválida - Aprobación de perfil. Información de contacto de mesa de entrada para consultar procedimiento a seguir disponible en el siguiente enlace: <https://www.fing.edu.uy/servicios/div-secretaria/secretaria>
 ...

Pregunta
 ¿Cuál es el proceso para efectuar un cambio de parcial?

Respuesta (Fine-tuning)
 Para efectuar un cambio de parcial, se debe **iniciar el trámite en la Mesa de Entrada** de la Facultad de Ingeniería. La información de contacto de esta mesa se encuentra disponible en el siguiente enlace: <https://www.fing.edu.uy/servicios/div-secretaria/secretaria>

Respuesta (RAG)
 Según el extracto de página web de título "Procedimientos para solicitudes" de la Facultad de Ingeniería, no se menciona específicamente el proceso para efectuar un cambio de parcial. Sin embargo, se puede consultar el procedimiento para solicitudes de excepción de cursos, que se tramitan hasta el 29/3/2023 para el primer semestre y hasta el 01/09/2023 para el segundo semestre.

Es importante mencionar que **no se cuenta con información para responder la pregunta sobre el proceso específico para efectuar un cambio de parcial.**

Figura 5.8: Ejemplo de pregunta con las respuestas generadas por la versión con fine-tuning y sin fine-tuning de Llama 3 8B.

a partir de la información provista en los *chunks*).

Por último, otra observación que se desprendió del análisis cualitativo es que los modelos aprendieron correctamente a responder que no cuentan con información cuando las preguntas son fuera del dominio. Si bien todos los ejemplos del dataset de evaluación son relacionados con el dominio, se hicieron algunas pruebas manuales con preguntas no relacionadas y se observó que correctamente el modelo respondía, casi de forma literal, la misma respuesta que fue utilizada en los ejemplos de preguntas fuera del dominio del dataset sintético.

Aun así, contrario a lo esperado, esto no ayudó a mejorar los casos en que no se cuenta con información en los documentos pero la pregunta sí está relacionada con el dominio, como se analizó anteriormente (ambos ejemplos en las Figuras 5.7 y 5.8 cumplen con esto).

Evaluación de los clasificadores de preguntas fuera del dominio

A continuación se presenta la metodología de evaluación y los resultados obtenidos en el experimento que fue explicado en la Sección 4.2.1, que consiste en entrenar clasificadores a partir del dataset sintético, para determinar si una pregunta está relacionada con el dominio de los documentos o es una pregunta fuera de dominio (recordar categorías de preguntas de la Sección 4.2.1).

Para realizar una evaluación justa de los clasificadores entrenados, se utilizó el dataset de 135 preguntas reales (véase Sección 5.1), las cuales se considera que pertenecen todas al dominio, y se utilizó el dataset MLQA (P. Lewis, Oguz, Rinott, Riedel, y Schwenk, 2020) provisto por Meta, que consiste en pares de preguntas y respuestas generados a partir de Wikipedia, en varios lenguajes entre los que se incluye el español. Estas preguntas son de diversos dominios, y debido a la naturaleza de nuestro caso de uso, podemos estar bastante seguros

que no estarán relacionadas con nuestro dominio.

Se seleccionaron 135 preguntas aleatorias en español del dataset MLQA, las cuales serán consideradas como preguntas fuera del dominio. Junto a las 135 preguntas del dataset de preguntas reales, se conformó un dataset de 270 instancias de preguntas y su clasificación (1 si es fuera del dominio y 0 si no).

Notar que lo único que se utilizó para entrenar los clasificadores fueron los documentos, a partir de los cuales se generó el dataset sintético. Las 135 preguntas reales no fueron utilizadas en ningún momento del pipeline de entrenamiento de los clasificadores (ni ninguna instancia del dataset MLQA).

La Tabla 5.5 presenta los resultados obtenidos con los clasificadores k-NN, SVM y regresión logística utilizando todo el dataset sintético y el dataset balanceado (según se explico en la Sección 4.2.1) para el entrenamiento. Se puede observar que tanto SVM como regresión logística con el dataset balanceado presentan resultados casi perfectos, con solo dos ejemplos mal clasificados. La Figura 5.9 presenta las matrices de confusión para estos dos clasificadores. Además, todos los métodos mejoran al usar el dataset sintético balanceado, especialmente regresión logística.

Clasificador	P Pos	R Pos	F1 Pos	P Neg	R Neg	F1 Neg	Acc
k-NN completo	1.00	0.43	0.60	0.64	1.00	0.78	0.71
SVM completo	1.00	0.95	0.97	0.95	1.00	0.97	0.97
Regresión log comp	1.00	0.47	0.64	0.65	1.00	0.79	0.73
k-NN balanceado	1.00	0.60	0.75	0.71	1.00	0.83	0.80
SVM balanceado	0.99	0.99	0.99	0.99	0.99	0.99	0.99
Regresión log bal	1.00	0.99	0.99	0.99	1.00	0.99	0.99

Tabla 5.5: Precision, Recall y F1 Score para las clases positiva y negativa y Accuracy, para los diferentes clasificadores

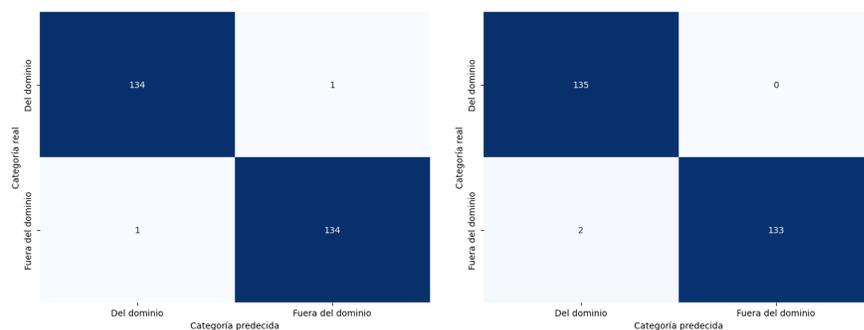


Figura 5.9: Matrices de confusión para los clasificadores de SVM (izquierda) y Regresión Logística (derecha) utilizando el dataset sintético balanceado.

En conclusión, los resultados obtenidos con este método son muy buenos y muestran el potencial de la construcción de datasets sintéticos para el entrenamiento de este tipo de clasificadores que pueden ser utilizados como *gates* al principio del pipeline de un sistema QA.

5.2.3. Prompting de derivaciones

Analizando los resultados obtenidos con el método de prompting de derivaciones, se puede observar que este método aumenta considerablemente la proporción de respuestas aceptables respecto al resto de los métodos. La Figura 5.10 muestra las gráficas de frecuencia de scores y proporción de respuestas aceptables/inaceptables para los modelos **Claude**, comparando los experimentos de RAG y los de derivaciones (con $k = 3$). Se presentan los resultados tanto utilizando el modelo *Bi-Encoder* como el modelo *Cross-Encoder* para la etapa de Retrieval.

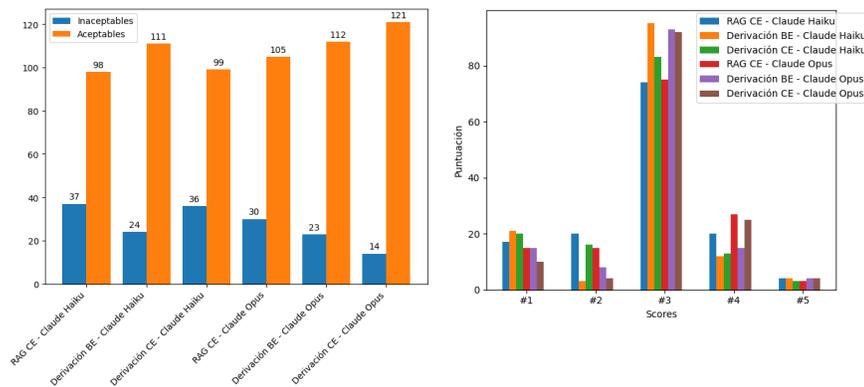


Figura 5.10: Gráficas correspondientes a experimentos usando los métodos RAG y Derivaciones con los modelos **Claude Haiku** y **Claude Opus**. Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

Sorprendentemente, los resultados de **Claude Haiku** utilizando el *Bi-Encoder* son mejores a los resultados utilizando el *Cross-Encoder*. De todas maneras, esto parece ser una excepción a la regla, ya que los experimentos con **Claude Opus** presentan el mismo comportamiento que los de RAG con respecto a la etapa de Retrieval, dando mejores resultados al usar el modelo *Cross-Encoder*.

Centrando el análisis en la comparación de los métodos RAG y Derivaciones, en particular para el modelo **Claude Opus**, como ya se comentó, hay una reducción considerable de respuestas inaceptables. Sin embargo, no hay un aumento en la cantidad de respuestas con scores de 4 y 5. Esto se debe a que muchas de las respuestas inaceptables de los experimentos con RAG reciben un score de 3 con prompting de derivaciones. Dos posibles explicaciones para este comportamiento son:

1. La regla NoInfo tiene mayor impacto que simplemente instruir al modelo a no responder preguntas cuando la información no está disponible (como se hace en RAG).
2. Generar respuestas a través de la aplicación de reglas restrictivas explícitamente definidas reduce la probabilidad de alucinaciones o malas interpretaciones de los *chunks* de contexto y minimiza el potencial de razonamientos erróneos, ya que delimita la tarea a un marco de acción más preciso.

A partir de lo anterior, podemos concluir que este método, en el caso de **Claude Opus**, no tiene prácticamente impacto en el Recall (i.e., responder la mayor cantidad de respuestas posibles) y aumenta considerablemente la Precision (i.e., evitar generar respuestas incorrectas). Esto se puede corroborar observando los resultados obtenidos con BERTScore, en donde los experimentos de derivaciones suelen aumentar el valor de Precision respecto a sus contrapartes en RAG.

Es importante resaltar que los resultados obtenidos con Claude Haiku no muestran exactamente los mismos comportamientos. Por un lado, se observa que también hay una reducción de las respuestas inaceptables, sobre todo en el experimento con *Bi-Encoder*. Sin embargo, a diferencia de los experimentos con **Claude Opus**, en estos sí se observa una disminución en la cantidad de respuestas con scores de 4 y 5, comparando con RAG. Esto sugiere que el tamaño del LLM es un factor relevante para resolver correctamente esta tarea. Modelos más grandes y capaces, como **Claude Opus**, muestran un mejor entendimiento de la tarea de construir la derivación, y son más capaces de aplicar las reglas de forma efectiva, llegando a mejores resultados.

Question:

Si realicé 5° año de biología de secundaria, ¿Puedo anotarme en la facultad de ingeniería sin repetir el año?

Derivation:

Futuros estudiantes Facultad de Ingeniería	INSCRIPCIÓN A CARRERA Facultad de Ingeniería
Cada carrera tiene un listado de Estudios preuniversitarios requeridos. [Extract]	Fórmula 69B para los estudiantes que se inscribirán en forma provisoria por tener una materia previa de secundaria. [Extract]
Cada carrera tiene un listado de Estudios preuniversitarios requeridos. Si tienes una materia previa de secundaria, puedes inscribirte en forma provisoria con la Fórmula 69B. [Compose]	
Cada carrera tiene un listado de Estudios preuniversitarios requeridos. Si tienes una materia previa de secundaria, puedes inscribirte en forma provisoria con la Fórmula 69B. [Refine]	
Si completaste 5° de biología de secundaria sin materias pendientes, entonces cumples con los estudios preuniversitarios requeridos y puedes anotarte en la Facultad de Ingeniería sin tener que repetir el año. En caso de que tuvieras una materia previa, podrías inscribirte en forma provisoria con la Fórmula 69B.	

Figura 5.11: Ejemplo de una derivación incorrecta. En la aplicación de la regla Refine, el modelo inventa que es suficiente completar 5to año de liceo en Biología para anotarse a la Facultad (alucinación subrayada en rojo).

Otra observación interesante, que surge de un análisis cualitativo de las respuestas y las derivaciones generadas, es la ganancia en interpretabilidad respecto a RAG. Esto se observa especialmente en los ejemplos que aún son evaluados con scores de 1 y 2. Para estos casos, el hecho de contar con el árbol de derivación facilita considerablemente la identificación de los errores realizados por el LLM

(en la aplicación de las reglas). Comparado con RAG, esto es una ventaja interesante ya que en muchos casos elimina la necesidad de verificar las respuestas teniendo que consultar directamente las fuentes. En cambio, alcanza con seguir el razonamiento de la derivación para identificar los pasos erróneos. La Figura 5.11 presenta un ejemplo real de una derivación incorrecta. En la aplicación de la regla Refine, es claro que el LLM produce alucinaciones no presentes en la hipótesis (el nodo hijo que se encuentra encima).

Comparación de los dos mejores experimentos

Los experimentos que obtuvieron mejores resultados de todos los métodos explorados fueron:

- Prompting de derivaciones con $k = 7$ usando el modelo de Retrieval *Cross-Encoder* y el LLM *Claude Opus*.
- RAG con $k = 7$ usando el modelo de Retrieval *Cross-Encoder* y el LLM *Claude Opus*.

Comparando los resultados obtenidos en estos dos experimentos, se observan puntos a favor y puntos en contra de cada uno.

En primer lugar, el experimento de derivaciones logra disminuir las respuestas inaceptables a 14, y obtiene la menor cantidad de respuestas con score 1 (9). En cambio, el experimento de RAG obtiene 23 respuestas inaceptables y 15 respuestas con score 1.

Sin embargo, mirando las respuestas con scores 4 y 5, quien obtiene mejores resultados es el experimento de RAG, con 41 respuestas calificadas de esa manera, además de contar con la mayor cantidad de respuestas de score 5 (9). Por otro lado, el experimento de derivaciones obtiene 28 respuestas con score 4 y 5, y 6 respuestas con score 5.

Entonces, se puede concluir que la elección del método más efectivo se resume a decidir entre minimizar la cantidad de respuestas inaceptables e incorrectas o maximizar la cantidad de respuestas útiles. Si el primer caso es más importante (por ejemplo, en casos de uso donde la información es delicada y una respuesta incorrecta no es aceptable) entonces el método de derivaciones ofrece mayor seguridad y precisión en las respuestas. Si el segundo caso es más importante (por ejemplo, en casos de uso en donde el sistema será usado por expertos para facilitar su tarea, y son capaces de detectar las respuestas incorrectas) entonces el método de RAG resultará más útil.

5.2.4. Grandes ventanas de contexto

Por último, se analizan los resultados obtenidos en los experimentos correspondientes al método que consiste en aprovechar las grandes ventanas de contexto. Para este experimento, resulta especialmente interesante la comparación con el método RAG. En la Sección 4.4 se planteó la pregunta de si, a medida

que se cuenta con LLMs con grandes ventanas de contexto, sigue siendo necesario seguir una estrategia de tipo RAG. Buscaremos responder esta pregunta en el caso particular de nuestro caso de uso, y con los modelos con los que se experimentó (Claude Haiku y Claude Opus).

La Figura 5.12 muestra las gráficas de frecuencia de scores y proporción de respuestas aceptables/inaceptables para el modelo Claude Opus, comparando con los experimentos de RAG y derivaciones, y la Figura 5.13 presenta las gráficas para el modelo Claude Haiku.

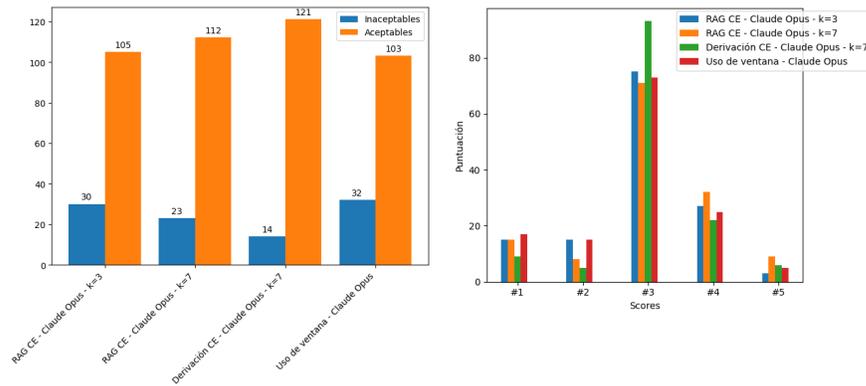


Figura 5.12: Gráficas correspondientes a experimentos usando los métodos RAG, Derivaciones y uso de ventanas con el modelo Claude Opus. Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

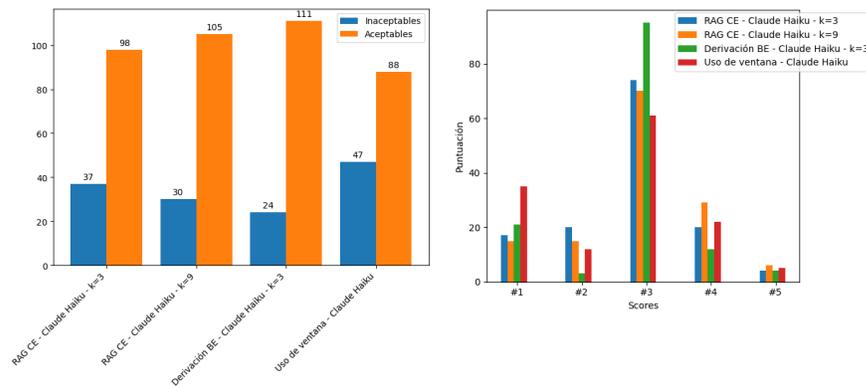


Figura 5.13: Gráficas correspondientes a experimentos usando los métodos RAG, Derivaciones y uso de ventanas con el modelo Claude Haiku. Izquierda: Cantidad de inaceptables y aceptables por experimento. Derecha: Frecuencia de los scores.

Observando los resultados obtenidos con los dos LLMs, hay una disminución

considerable de respuestas consideradas aceptables respecto a RAG (y derivaciones). Una posible explicación para esto puede ser que la cantidad de información no relevante del contexto (la enorme mayoría de la información del contexto es no relevante), tienda a aumentar las respuestas con información incorrecta, similar a lo que sucedía con `Mistral 7B` al aumentar el valor de k en RAG.

Centrando la atención en los resultados de `Claude Opus`, podemos ver que son muy similares a los resultados del método RAG con $k = 3$, y considerablemente inferiores a los resultados del método RAG con $k = 7$. En particular, hay una disminución de respuestas con scores de 4 y 5 (30 frente a 41) y un aumento de respuestas inaceptables (32 frente a 23). Estos resultados muestran claramente que la etapa de Retrieval es aún importante, y el uso del gran tamaño de la ventana de contexto del modelo `Claude Opus` para evitar esta etapa terminó resultando perjudicial.

Esta observación es aún más notoria en los experimentos usando `Claude Haiku`. En este caso, el aumento de respuestas con score 1 es muy considerable, reduciendo en gran medida las respuestas con score 3. Parece haber una dificultad acentuada en la tarea de detectar los casos en los que ninguna información es relevante y es necesario responder que no se cuenta con información. Esto podría indicar que los modelos más chicos tienen aún más dificultades en la utilización efectiva de sus ventanas de contexto.

En conclusión, de los experimentos realizados se puede concluir que, al menos en la configuración de nuestros experimentos, el aumento en el tamaño de la ventana de contexto no sustituye a la etapa de Retrieval y los métodos tipo RAG. Además, como era de esperarse, el tamaño del LLM tiene un impacto considerable al utilizar la ventana de contexto con todos los documentos, ya que `Claude Opus` cometió bastantes menos errores que `Claude Haiku`.

5.3. Evaluación humana

Debido a la evidenciada dificultad de evaluar automáticamente los experimentos de forma fiable, realizar una evaluación manual por anotadores humanos resulta importante para validar el análisis, así como los propios métodos de evaluación utilizados. En particular, nos propusimos intentar medir el acuerdo de los evaluadores humanos con el método de evaluación utilizando el LLM `Claude Opus`.

Para realizar la evaluación humana, se implementó una herramienta que obtuviera desde archivos CSV las respuestas generadas con cada método y permitiera evaluar utilizando el mismo criterio que se definió en la Sección 5.1.2 para la evaluación con LLM (escala del 1 al 5). Cada anotador debe ingresar a la aplicación web, seleccionar el o los métodos a evaluar, ingresar su nombre, y las respuestas generadas se van desplegando en orden a medida que el anotador va realizando las evaluaciones.

La anotación llevada a cabo consistió en dos anotadores humanos (la Anotadora 1 fue la tutora Aiala Rosá y el Anotador 2 fue el estudiante Ignacio Sastre), y se decidió evaluar el método de prompting de derivaciones con `Claude Opus`

y $k = 3$. La Anotadora 1 evaluó 66 instancias del dataset de preguntas reales, y el Anotador 2 evaluó 94 instancias. Adicionalmente, la Anotadora 1 también realizó 65 evaluaciones del método de prompting de derivaciones con **Claude Haiku** para comparar ambos modelos.

La Tabla 5.6 presenta los resultados de la evaluación realizada por la Anotadora 1 sobre 65 instancias del dataset para los experimentos mencionados anteriormente. Se pueden realizar dos observaciones interesantes de esta comparación:

- A diferencia de los resultados obtenidos con la evaluación automática, los resultados de la evaluación humana sobre el método de derivaciones con **Claude Opus** y **Claude Haiku** son muy similares.
- Incluso, **Claude Haiku** obtiene una mayor proporción de respuestas aceptables que **Claude Opus**, aunque **Claude Opus** obtiene levemente un mayor promedio de los scores.

Aun así, es importante notar que la evaluación automática se realizó sobre todas las instancias del dataset (135), por lo que la comparación no es perfecta.

Método	k	LLM	#1	#2	#3	#4	#5	Avg	N avg	Stdev	Acceptables
Derivación BE	3	Claude Haiku	8	7	27	20	3	3.05	0.51	1.05	0.769
Derivación CE	3	Claude Opus	7	10	25	17	6	3.08	0.52	1.11	0.738

Tabla 5.6: Resultados de los experimentos de prompting de derivaciones con las evaluaciones realizadas por la Anotadora 1 sobre 65 instancias del dataset de preguntas reales.

Algunas apreciaciones cualitativas que fueron realizadas por los evaluadores humanos son las siguientes:

- Hay algunos casos en donde las respuestas generadas no son incorrectas (no contradicen la respuesta de referencia), pero aportan información distinta que también es válida y podría ser útil. Este tipo de situaciones son imposibles de apreciar correctamente por la evaluación automática que solo tiene en cuenta la respuesta de referencia.
- En otros casos, las respuestas generadas pueden ser demasiado generales y no responder las particularidades de la pregunta, incluso agregando información irrelevante.
- Existen algunas preguntas en el dataset que no son muy adecuadas para este tipo de sistemas, ya que requieren de la experiencia de quien responde y no es información que se encuentre disponible en documentos.

Para evaluar de forma más precisa el método de evaluación automática utilizando el LLM, se restringió el análisis a las 66 evaluaciones que fueron realizadas por ambos anotadores humanos sobre el método de derivaciones con **Claude Opus**. La Tabla 5.7 presenta los resultados de los tres evaluadores (los dos humanos y **Claude Opus**).

Anotador	#1	#2	#3	#4	#5	Avg	N avg	Stdev	Aceptables
Anotadora 1	7	10	25	18	6	3.09	0.52	1.11	0.742
Anotador 2	9	10	24	16	7	3.03	0.51	1.18	0.712
Claude Opus	6	2	43	13	2	3.05	0.51	0.85	0.879

Tabla 5.7: Resultados del experimento de derivaciones con Claude Opus, $k = 3$, evaluado por los anotadores humanos y Claude Opus.

Se puede observar que la evaluación automática tiene bastantes más evaluaciones con score 3 que las evaluaciones humanas. Esto lleva a una disminución en las evaluaciones inaceptables (scores 1 y 2), por lo que la proporción de respuestas aceptables es superior según la evaluación automática. También hay una disminución en las evaluaciones de scores 4 y 5 (especialmente 5), lo que impacta en el promedio de scores (la Anotadora 1 obtiene el mayor promedio).

Por otro lado, se realiza un análisis para determinar el acuerdo entre anotadores. Para ello, se utilizan los siguientes coeficientes:

1. **Coefficiente de correlación de Spearman:** Es una medida de la correlación entre dos variables aleatorias, que mide la dependencia estadística entre los rangos de las dos variables (Spearman, 1904). Este coeficiente solo considera el orden relativo de las evaluaciones, y dará un valor alto cuando estas tengan un orden relativo similar. Varía entre -1 y 1, y decimos que hay una correlación negativa si el valor es menor a 0, y una correlación positiva si el valor es mayor que 0.
2. **Coefficiente Kappa de Cohen ponderado:** Es una medida de la concordancia entre dos muestras (Cohen, 1960). Mientras que el Kappa de Cohen no ponderado se utiliza para variables nominales, la versión ponderada es adecuada para variables ordinales, en donde existe un orden entre los valores. A diferencia de Spearman, este coeficiente tiene en cuenta las magnitudes de los valores, y menores diferencias penalizan menos que mayores diferencias. Al igual que Spearman, varía entre -1 y 1.
3. **Coefficiente Kappa de Fleiss:** Es una medida de la concordancia entre múltiples muestras (Fleiss, 1971), a diferencia de Kappa de Cohen, que solo mide la concordancia entre dos muestras. También varía entre -1 y 1.

El resultado obtenido con el coeficiente Kappa de Fleiss sobre las evaluaciones de los tres anotadores es de **0.377**. La Tabla 5.8 presenta los resultados obtenidos para las distintas combinaciones de anotadores, de los primeros dos coeficientes. Adicionalmente, se presentan los resultados obtenidos sobre las 94 instancias evaluadas por el Anotador 2.

Se observa un acuerdo moderado entre los anotadores humanos, así como entre el Anotador 2 y el LLM. Sin embargo, el acuerdo entre la Anotadora 1 y el LLM es más débil. La correlación de Spearman entre el Anotador 2 y el LLM es levemente superior a la correlación entre los evaluadores humanos, pero sucede al revés con el coeficiente Kappa de Cohen. Debido a que Spearman solo considera el orden relativo de las evaluaciones, esto podría indicar que el LLM

Anotadores	# Instancias	Spearman	Cohen Kappa	F1 (Acep.)
1 y 2	66	0.578	0.563	0.809
1 y Claude Opus	66	0.282	0.283	0.630
2 y Claude Opus	66	0.587	0.457	0.697
2 y Claude Opus	94	0.573	0.439	0.691

Tabla 5.8: Correlación de Spearman y Cohen Kappa ponderado entre las evaluaciones de las distintas combinaciones de anotadores, y F1 Score de las evaluaciones clasificadas en aceptables e inaceptables.

podría estar captando bastante bien el orden relativo de las evaluaciones del Anotador 2, pero estaría asignando calificaciones que difieren más en magnitud que las diferencias entre los anotadores humanos, lo que explicaría el Cohen de Kappa ponderado más bajo.

Además de medir los coeficientes, se calcula el F1 Score para las categorías de respuestas aceptables (scores 3, 4 y 5) e inaceptables (scores 1 y 2). Se observa que hay bastante mayor alineación entre los anotadores humanos sobre este aspecto que con el LLM. Esta discrepancia puede explicarse por la gran cantidad de evaluaciones con score 3 de la evaluación automática, lo que lleva a un aumento de respuestas consideradas aceptables.

De lo anterior se concluye que:

- El acuerdo moderado entre los evaluadores humanos demuestra la dificultad de la tarea de evaluar las respuestas.
- Existe un acuerdo variable entre la evaluación realizada por el LLM y las evaluaciones realizadas por humanos. El LLM se alinea mejor con los criterios utilizados por el Evaluador 2.
- Queda en evidencia que la evaluación automática dista mucho de ser perfecta. Aun así, en términos relativos, el LLM parece capturar en varios casos el orden relativo de las evaluaciones humanas, lo que da sustento a los análisis realizados en la sección anterior.

Por último, se presentan las matrices de confusión de las evaluaciones realizadas, para cada combinación de evaluadores, tanto para todos los scores, como para la agrupación en aceptables e inaceptables. La Figura 5.14 corresponde a los anotadores humanos, la Figura 5.15 compara a la Anotadora 1 con Claude Opus, la Figura 5.16 compara al Anotador 2 con el LLM y la Figura 5.17 compara a estos mismos evaluadores pero con 94 instancias.

5.4. Síntesis

Es este capítulo se definió la metodología de evaluación, se presentaron los resultados obtenidos y se realizó un análisis detallado de estos resultados. Se

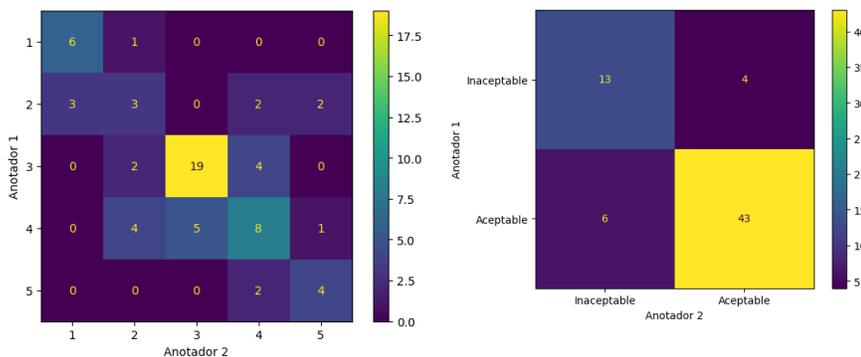


Figura 5.14: Izquierda: Matriz de confusión de las evaluaciones realizadas por los anotadores 1 y 2. Derecha: Matriz de confusión de las mismas evaluaciones agrupadas en aceptables e inaceptables.

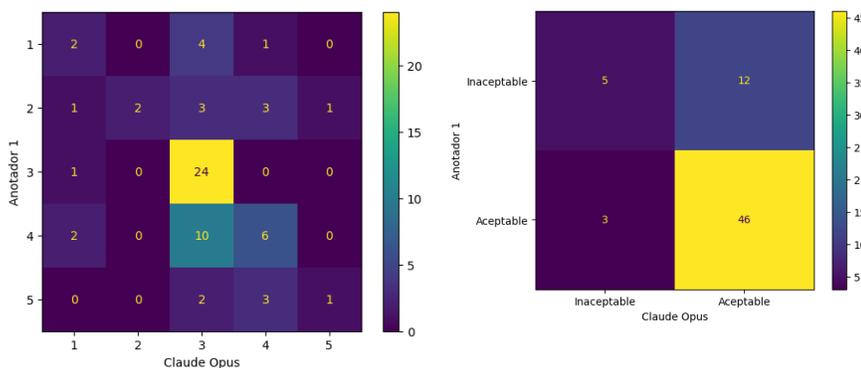


Figura 5.15: Izquierda: Matriz de confusión de las evaluaciones realizadas por los anotadores 1 y Claude Opus. Derecha: Matriz de confusión de las mismas evaluaciones agrupadas en aceptables e inaceptables.

observó que el método de prompting de derivaciones es el que obtiene mejores resultados en términos de minimizar las respuestas inaceptables. Por otro lado, el método de RAG es el que obtiene mejores resultados en términos de maximizar las respuestas útiles.

Se realizó una evaluación humana para validar los resultados obtenidos con la evaluación automática. Se observó un acuerdo moderado entre los anotadores humanos, y un acuerdo variable entre los anotadores humanos y el LLM, obteniendo un acuerdo mayor con uno de los anotadores.

En el siguiente capítulo se describe el diseño y construcción del prototipo de la herramienta, que se basa en los resultados obtenidos en este capítulo.

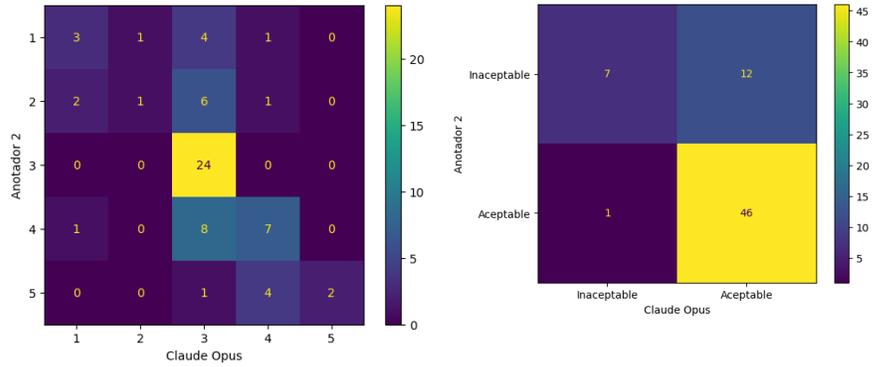


Figura 5.16: Izquierda: Matriz de confusión de las evaluaciones realizadas por los anotadores 2 y Claude Opus. Derecha: Matriz de confusión de las mismas evaluaciones agrupadas en aceptables e inaceptables.

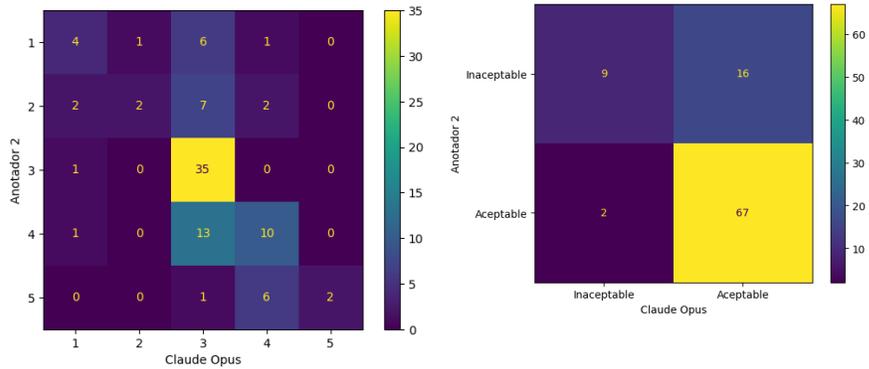


Figura 5.17: Izquierda: Matriz de confusión de las evaluaciones realizadas por los anotadores 2 y Claude Opus, sobre 94 instancias. Derecha: Matriz de confusión de las mismas evaluaciones agrupadas en aceptables e inaceptables.

Capítulo 6

Construcción del prototipo

Una vez finalizada la experimentación, y con los resultados de la evaluación analizados, fue posible tomar las decisiones de diseño necesarias para la construcción de un prototipo del sistema.

El objetivo de este prototipo es poner a disposición la herramienta para los integrantes del Espacio de Orientación y Consulta (EOC), para obtener retroalimentación del funcionamiento y realizar una evaluación más real, pero sin dar acceso aún a los usuarios finales (quienes realizan las consultas directamente).

Se definieron los siguientes requerimientos funcionales:

- **Realizar una consulta:** Funcionalidad principal que debe permitir al usuario ingresar una pregunta, para generar una respuesta a dicha pregunta que además indique las fuentes de información utilizadas.
- **Evaluar las respuestas:** Los usuarios deben poder evaluar las respuestas generadas, de forma positiva o negativa, y opcionalmente ingresar un comentario sobre la respuesta.
- **Ver preguntas y respuestas anteriores:** Los usuarios deben poder ver las preguntas ya realizadas junto con las respuestas que fueron generadas.
- **Carga de documentos:** Debe ser posible cargar documentos nuevos al sistema para ser utilizados como fuentes de información en la generación de respuestas.

En este capítulo se describen las decisiones tomadas a partir de la experimentación, así como el diseño de la arquitectura y la interfaz de usuario de la herramienta desarrollada.

6.1. Decisiones a partir de la experimentación

En función de lo visto en la Sección 5.2 de resultados, y dadas las limitaciones de recursos, se tomaron las siguientes decisiones:

- **Método:** Se decidió utilizar por defecto el método de **prompting de derivaciones**. A partir de la comparación que se realiza en la Sección 5.2.3 entre el método RAG y derivaciones, se decidió ir por la opción que minimiza las respuestas inaceptables, haciendo más segura la herramienta en detrimento de maximizar la cantidad de respuestas útiles. Aun así, se diseñó la herramienta para que sea posible elegir otros métodos (RAG y uso de ventana de contexto) como opción de configuración.
- **Modelo de generación:** Debido a que los recursos con los que se realizaron los experimentos son parte del cluster *Cluster.uy*, y no es posible utilizar dichos recursos para ofrecer servicios, no es posible contar con una infraestructura local para desplegar modelos abiertos. A partir de esta limitación, se decidió utilizar los modelos cerrados a través de APIs. En particular, dado los costos de consumo, se optó por utilizar por defecto el modelo **Claude 3 Haiku**, que se estima en un gasto de menos de 400 pesos uruguayos por año asumiendo 10 consultas por día. De todas formas, se diseñó la herramienta para que también sea posible elegir entre múltiples modelos de múltiples proveedores (Anthropic, Google, OpenAI).
- **Modelo de Retrieval:** Nuevamente, debido a las restricciones de recursos, se optó por utilizar modelos **Bi-Encoder** disponibles por API, que generan *sentence embeddings* que luego son almacenados en una base de datos propia. En particular, se decidió utilizar por defecto el modelo **text-embedding-3-small** de OpenAI.

6.2. Arquitectura del prototipo

El prototipo sigue una arquitectura cliente-servidor, donde el servidor cuenta con una capa de persistencia (base de datos PostgreSQL) y una capa lógica, que ofrece una RESTful API (implementada en Python con el framework Django). Esta API es consumida por el cliente, que se trata de una aplicación web desarrollada en React con el framework Next.js. La Figura 6.1 muestra el diagrama de componentes de la arquitectura diseñada.

La capa lógica cuenta con dos componentes, que corresponden a distintas aplicaciones dentro del mismo proyecto Django:

- **Módulo QA:** Encargado de manejar las implementaciones de los diferentes métodos (Derivaciones, RAG, etc), así como la comunicación con las APIs de los modelos de generación. Además, se encarga del manejo de las preguntas y respuestas, así como registrar las evaluaciones realizadas por los usuarios.
- **Módulo Documents:** Encargado del manejo de los documentos, los métodos de Retrieval (*k-Nearest Neighbors*), así como la comunicación con las APIs de los modelos de Retrieval.

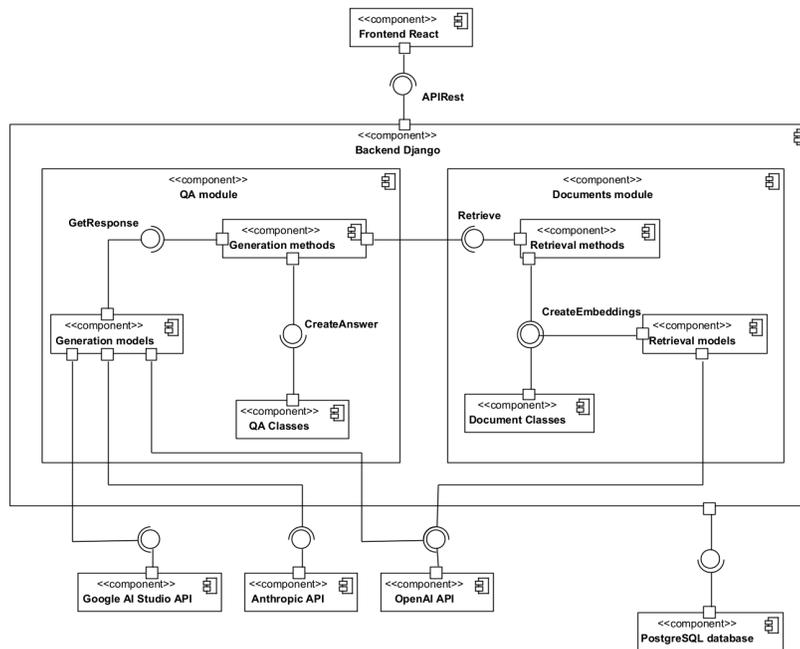


Figura 6.1: Diagrama de componentes del prototipo desarrollado.

6.2.1. Módulo QA

La Figura 6.2 presenta el diagrama de clases del módulo QA. Este módulo cuenta con tres clases:

- **Question:** Las preguntas realizadas por los usuarios.
- **Answer:** Las respuestas generadas para una pregunta. A una misma pregunta le pueden corresponder múltiples respuestas, para poder comparar distintas estrategias de generación. Si el método es derivaciones, se guarda un diccionario con el árbol de derivación generado. Además, se asocian a la respuesta los *chunks* recuperados en la etapa de Retrieval.
- **Evaluation:** Registro de evaluaciones realizadas por los usuarios de las respuestas generadas. Se puede votar positivamente o negativamente (atributo *like*), y de forma opcional, se puede registrar un comentario y el nombre del autor de la evaluación.

Además, en este módulo se encuentran las implementaciones de cada método (RAG, derivaciones y uso de grandes ventanas de contexto). Esto fue desarrollado de forma tal que sea sencillo agregar nuevos métodos. Cada método se implementa en un archivo nuevo, y debe implementar una función con el siguiente cabezal: `create_answer(message, model, temperature)`.

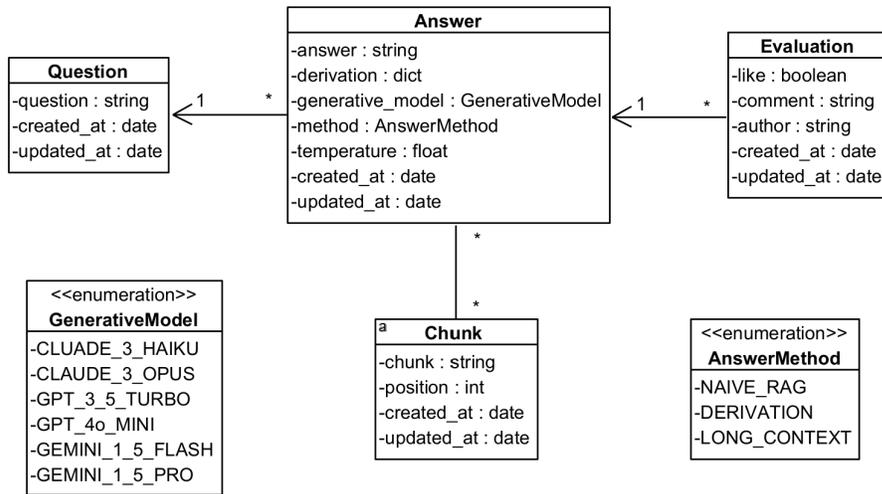


Figura 6.2: Diagrama de clases del módulo QA. La clase *Chunk* pertenece al módulo Documents, pero se agrega para mostrar la relación con *Answer*.

Esta función es la encargada de generar la respuesta siguiendo el método que corresponda, y para eso dispone de las funciones `get_response` (encargada de ejecutar el LLM) y `retrieve` (encargada de realizar la etapa de Retrieval).

Por último, en este módulo también se encuentra la comunicación con las APIs de los LLMs (Anthropic, Google y OpenAI). Al igual que con los métodos, se siguió una estrategia en donde la implementación de la comunicación con cada API es un archivo diferente, con la responsabilidad de implementar la función: `get_response(system_prompt, messages, model_id, temp, max_tok)`.

Esta función debe retornar la respuesta del LLM a partir de los parámetros especificados. De esta forma, es sencillo agregar nuevos proveedores. Además, para agregar nuevos modelos de los proveedores ya implementados, solamente es necesario agregar el identificador del modelo al enumerado *GenerativeModel* (véase Figura 6.2).

6.2.2. Módulo Documents

La Figura 6.3 presenta el diagrama de clases del módulo Documents. Este módulo cuenta con tres clases:

- **Document:** Los metadatos de los documentos que forman parte de la base de documentos usados para la etapa de Retrieval. Se cuenta con el parámetro *active* que permite habilitar o deshabilitar los documentos para ser usados como fuentes de información.
- **Chunk:** Los *chunks* que componen a los documentos. El contenido de cada

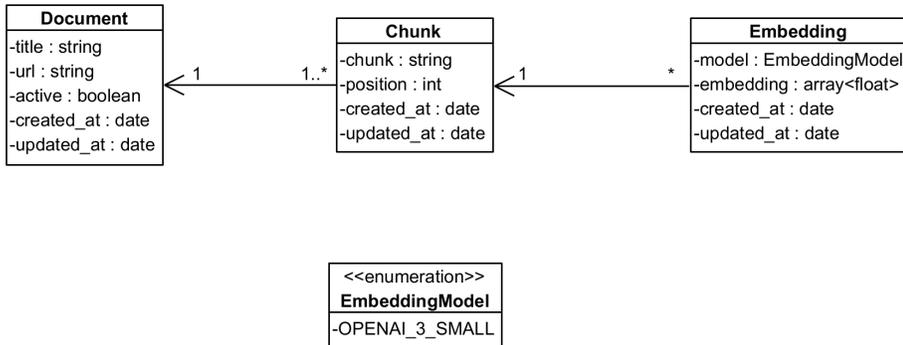


Figura 6.3: Diagrama de clases del módulo Documents.

documento se divide en un conjunto de *chunks*. Se mantiene la posición relativa de cada *chunk* en el documento.

- **Embedding:** Los vectores (*sentence embeddings*) asociados a cada *chunk*. Un *chunk* puede tener más de un vector asociado, para poder utilizar modelos *Bi-Encoder* distintos.

Para la carga de documentos, la clase *Documents* implementa el método estático `load_chunks_documents(csv_path)`, que a partir de la ruta a un archivo CSV que contiene los *chunks* asociados a un conjunto de documentos, junto a los metadatos de los documentos (título, url), crea las instancias de documentos y *chunks* que corresponda.

Por otro lado, para la generación de *embeddings* de los *chunks*, la clase *Chunks* implementa el método estático `create_embeddings(model, chunks)`, que genera y asocia los *embeddings* a cada *chunk* de la lista de *chunks* pasada por parámetro, utilizando un modelo particular.

Para cada proveedor de modelos *Bi-Encoder*, al igual que con los proveedores de LLMs, se cuenta con un archivo separado que debe implementar la función `get_embeddings(model_id, texts)`, en la cual se obtienen los *embeddings* para los textos utilizando la API específica de ese proveedor. Esta función es utilizada por el método de la clase *Chunk* descrito previamente.

Por último, este módulo es también responsable de implementar los métodos de Retrieval. Actualmente solo se encuentra implementado el método utilizando *k-Nearest Neighbors*, pero, al igual que en los métodos de generación, se permite de forma sencilla agregar otras implementaciones en archivos separados que deben contener la implementación de la función `retrieve(model, message, k)`, que retorna los *k chunks* más relevantes al mensaje.

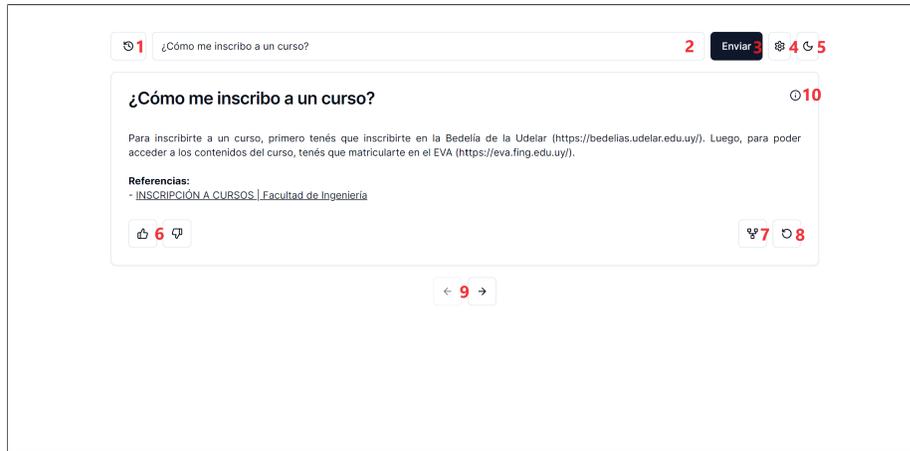


Figura 6.4: Captura de la pantalla principal del prototipo, con cada elemento interactivo numerado.

6.3. Interfaz de usuario

A continuación se presenta el diseño de la interfaz de usuario del prototipo. La Figura 6.4 muestra la pantalla principal. Cada elemento numerado en la figura se explica a continuación:

1. **Listado de preguntas anteriores:** Botón que despliega un modal con el listado de las preguntas ya realizadas, con la posibilidad de seleccionar una de ellas para ver la respuesta.
2. **Entrada para ingresar pregunta:** Entrada de texto para ingresar la pregunta.
3. **Enviar:** Botón para generar la respuesta a la pregunta.
4. **Configuración:** Botón que despliega un modal con las opciones de configuración para la generación de la respuesta, como la elección del método, modelo de generación y temperatura. La Figura 6.5 muestra el modal desplegado.
5. **Modo oscuro:** Botón para activar o desactivar modo oscuro.
6. **Me gusta/No me gusta:** Botones para iniciar el proceso de evaluación de una respuesta. Una vez seleccionado uno de los dos botones, se despliegan campos para ingresar un comentario y el nombre del autor (ambos opcionales). La Figura 6.6 muestra los campos desplegados.
7. **Derivación:** Botón que muestra el árbol de derivación generado para la respuesta, si la respuesta se generó con el método de prompting de derivaciones. La Figura 6.7 muestra un ejemplo de derivación.

8. **Regenerar:** Botón que permite regenerar la respuesta a la misma pregunta. Se genera con los parámetros definidos en la configuración.
9. **Cambiar respuesta:** Botones que permiten moverse entre las distintas respuestas generadas, en el caso en que hay más de una respuesta generada para una misma pregunta.
10. **Tooltip de información:** Tooltip que muestra la información de configuración de una respuesta ya generada.

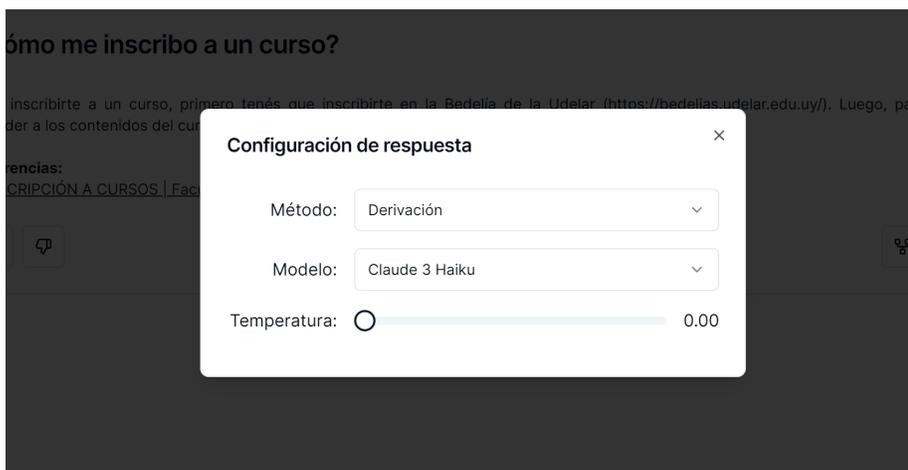


Figura 6.5: Captura de la pantalla de configuración de las respuestas del prototipo.

6.4. Síntesis

Se presentó el diseño y construcción del prototipo de la herramienta. Se describieron las decisiones tomadas a partir de la experimentación, así como la arquitectura y la interfaz de usuario del sistema desarrollado.

Este sistema se encuentra disponible para ser utilizado por los integrantes del Espacio de Orientación y Consulta, quienes están probando la herramienta y se espera obtener retroalimentación sobre su funcionamiento y utilidad. Luego de esta primera etapa de evaluación, en función del *feedback* obtenido, se considerará la posibilidad de dejar disponible la herramienta a los usuarios finales.

El siguiente capítulo presenta las conclusiones del proyecto, y las líneas que quedan abiertas para desarrollar o profundizar en trabajos futuros.

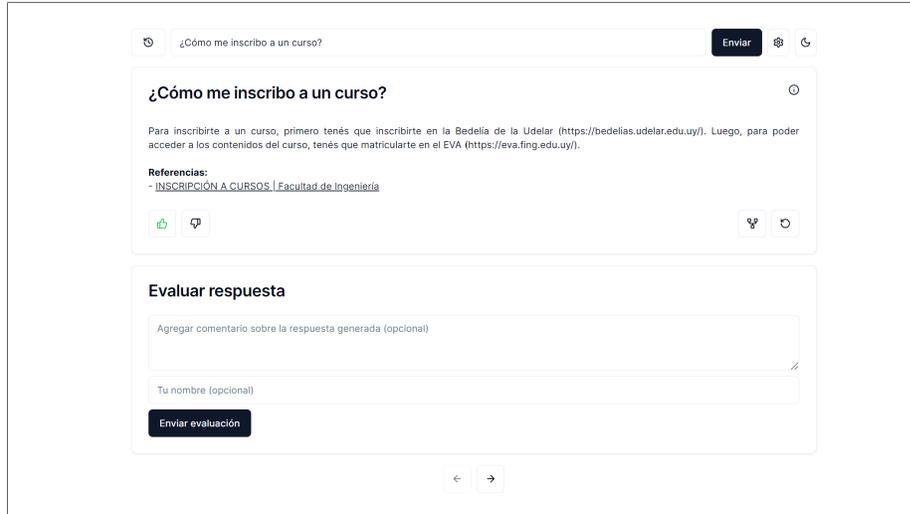


Figura 6.6: Captura de la pantalla principal del prototipo con la sección para evaluar respuestas expandida.

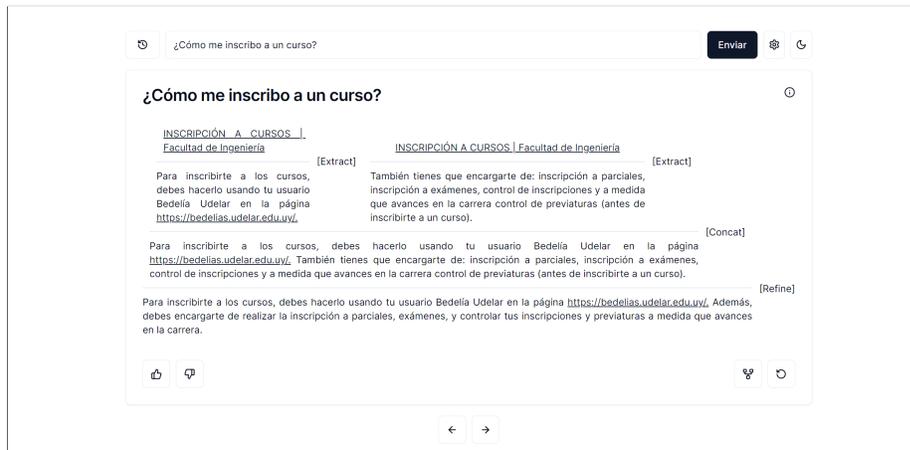


Figura 6.7: Captura de la pantalla principal del prototipo con el árbol de derivación de la respuesta generada.

Capítulo 7

Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones del proyecto, y las diferentes líneas que quedan abiertas para desarrollar o profundizar en trabajos futuros.

7.1. Conclusiones

Este proyecto consistió en explorar e implementar múltiples técnicas para acotar la generación de respuestas de los grandes modelos de lenguaje, en los casos en que se utiliza un conjunto acotado de documentos de un dominio particular como fuentes de información. En particular, se enfocó en un caso de estudio relacionado con el Espacio de Orientación y Consulta de Facultad de Ingeniería, para el cual se desarrolló una herramienta que permite responder consultas de estudiantes sobre su proceso académico.

Para cumplir con los objetivos definidos al inicio del documento, se realizaron múltiples experimentos con diferentes métodos: Retrieval-Augmented Generation, Fine-tuning de LLMs, Prompting de derivaciones y Grandes ventanas de contexto.

Además, se realizó una evaluación automática de cada uno de los experimentos realizados y se hizo un análisis de los resultados obtenidos. También se realizó evaluación humana, ya que se observó que las metodologías de evaluación automáticas en este tipo de tareas de generación de texto no son aún demasiado fiables.

Por último, se desarrolló una aplicación web que permite realizar consultas sobre la Facultad de Ingeniería, y obtener respuestas utilizando los distintos métodos mencionados. La herramienta se puso a disposición de los integrantes del EOC para evaluar su utilidad y obtener retroalimentación.

Se logró cumplir con los objetivos definidos y se obtuvieron los siguientes resultados:

- Se desarrolló un nuevo método al que se llamó Prompting de Derivaciones, que obtuvo resultados prometedores en el caso de estudio y ofrece algunas ventajas respecto al método de RAG. Este método fue aceptado en la conferencia *Iberamia 2024*¹ y será publicado por la editorial *Springer* (Sastre, Moncecchi, y Rosá, 2024).
- Se realizó un análisis exhaustivo comparando los diferentes métodos explorados, del cual se desprenden algunas observaciones no triviales:
 - La etapa de Retrieval en el método de RAG sigue siendo relevante, aún contando con modelos cuyas ventanas de contexto son suficientemente grandes como para evitar ese paso.
 - El método de Derivaciones obtiene la mayor proporción de respuestas aceptables, mientras que RAG maximiza las respuestas útiles.
 - El método de Fine-tuning seguido en este proyecto tuvo un impacto negativo, aumentando las alucinaciones y haciendo peor uso del contexto.
- Se midió el acuerdo entre evaluadores humanos y la evaluación realizada con un LLM, mostrando que existe un acuerdo moderado y que el modelo fue más eficaz en capturar el orden relativo de las evaluaciones que en asignar las calificaciones exactas según las anotaciones humanas.
- Se propuso un método de construcción de un clasificador de preguntas fuera de dominio utilizando únicamente un conjunto de documentos base sobre el dominio particular. Este método obtuvo resultados casi perfectos (0.99 F1 Score), mostrando la eficacia del método desarrollado.
- Se construyó un prototipo de herramienta que quedó disponible para ser utilizada por el EOC, que permite responder preguntas utilizando el método de Derivaciones propuesto.

7.2. Trabajo Futuro

A partir del trabajo realizado, surgen múltiples posibles mejoras a los experimentos y la metodología que se llevó a cabo, que se detallan a continuación.

7.2.1. Retrieval-Augmented Generation

Actualmente existe un desarrollo muy amplio de nuevas técnicas que mejoran el método de RAG tradicional (Gao y cols., 2024). Muchas de estas técnicas no fueron exploradas en este trabajo.

Por ejemplo, se podría explorar la posibilidad de realizar Fine-tuning sobre el modelo de Retrieval para mejorar la etapa de recuperación. También se podría explorar la utilización de técnicas de prompting como Chain of Thoughts (Wei

¹<https://www.iberamia.org/iberamia/iberamia2024/>

y cols., 2024), Tree of Thoughts (Yao, Yu, y cols., 2023), o ReAct (Yao, Zhao, y cols., 2023) en la etapa de generación.

7.2.2. Fine-tuning de LLM

Si bien los resultados de este experimento no fueron alentadores, es posible que el método empleado sea mejorable para mitigar los problemas encontrados.

Es posible que el dataset sintético introduzca errores que puedan estar afectando negativamente en el momento de realizar el fine-tuning. Una alternativa al dataset sintético, es construir manualmente un dataset de entrenamiento con respuestas reales, utilizando un conjunto de mails diferentes a los utilizados para evaluar. Este proceso no es sencillo ya que requiere de encontrar las fuentes de información de las respuestas a cada mail, pero permitiría comparar los resultados obtenidos con los dos tipos de datasets, e incluso combinarlos.

Tampoco se realizó una búsqueda de hiperparámetros para determinar los más adecuados para el método LoRA en el proceso del fine-tuning. Realizar un estudio más profundo y exhaustivo de este método podría derivar en resultados más competitivos con el resto de métodos.

7.2.3. Prompting de Derivaciones

Este método abre múltiples líneas de investigación que se podrían explorar.

En primer lugar, aunque el conjunto de reglas utilizado en este proyecto se diseñó de forma general, aborda específicamente las necesidades del caso de estudio llevado a cabo. Una posibilidad es explorar múltiples conjuntos de reglas diferentes, que permitan comparar los resultados obtenidos con las definidas en este proyecto, y también permitan adaptar este método a otros casos de estudio, e incluso otras tareas que no incluyan etapa de Retrieval (como tareas de razonamiento lógico o resolución de problemas matemáticos).

Además, se puede trabajar en definir el lenguaje formal que existe detrás de la aplicación de las reglas, así como explorar la posibilidad de desarrollar una metateoría que permita dar cierta validez a las derivaciones construidas. Por ejemplo, mediante el desarrollo de un lenguaje formal que permita construir fórmulas φ que representen las oraciones que se van generando mediante la construcción del árbol, y el desarrollo de una función de interpretación $f(\varphi)$ que utilice heurísticas para construir oraciones en lenguaje natural a partir de las fórmulas, podría ser posible añadir una capa de validación que compare las oraciones generadas por el LLM mediante la aplicación de las reglas de derivación con las oraciones obtenidas con la función $f(\varphi)$.

Otro camino posible a explorar es realizar fine-tuning de LLMs para la tarea particular de construir derivaciones. En este proyecto, se exploraron únicamente técnicas de prompting. Sin embargo, en trabajos como React (Yao, Zhao, y cols., 2023), se obtuvieron mejores resultados mediante fine-tuning que solamente con prompting. Se podría seguir un camino similar al método de fine-tuning explorado en este proyecto, que consista en construir un dataset sintético de derivaciones para utilizar en el entrenamiento.

Por último, utilizando el lenguaje formal y la función de interpretación que se propuso antes, se podría seguir una metodología similar a la propuesta por [Zelikman, Wu, Mu, y Goodman \(2022\)](#). En ese trabajo, se aplica un loop que consiste en: (1) construir un dataset sintético de ejemplos (en nuestro caso de derivaciones) con un LLM, (2) validar la correctitud de los ejemplos construidos (en nuestro caso se podría utilizar la función de interpretación como se describió anteriormente), y (3) realizar fine-tuning solamente con los ejemplos correctos, sobre el mismo LLM. Se aplican múltiples iteraciones de este loop buscando que el modelo se mejore a sí mismo.

7.2.4. Grandes ventanas de contexto

En este proyecto, se experimentó con introducir todos los documentos en la ventana de contexto del LLM. Este experimento podría ser visto como un caso particular de RAG, en el que la cantidad k de *chunks* recuperados es igual a la cantidad total de *chunks*. Por otro lado, se exploró con diferentes valores de k , pero no se realizó una búsqueda exhaustiva del mejor valor de k para los modelos Claude Haiku y Claude Opus. Si bien se mostró que RAG obtiene mejores resultados que introducir todos los documentos en la ventana de contexto, también se observó que el aumento de k daba mejores resultados para los modelos cerrados. Un posible trabajo futuro es realizar un estudio más exhaustivo variando el valor de k , desde 1 hasta la cantidad total de *chunks* (caso en el que se insertan todos los documentos).

Además, la forma en que los documentos se agregan en el contexto no fue explorada en profundidad. El fenómeno denominado “*lost in the middle*” ([Liu y cols., 2024](#)) sugiere que los modelos tienden a prestar más atención a los tokens que se encuentran al principio y al final de la ventana, en contraposición con los que se encuentran en el medio. Podría ser interesante explorar la posibilidad de ordenar los documentos en la ventana según su relevancia con la consulta, siguiendo algún criterio que respete este fenómeno.

7.2.5. Metodología de evaluación

La evaluación de las tareas que involucran generación de texto utilizando grandes modelos de lenguaje es un tema fundamental que, como se evidenció en este proyecto, aún carece de una metodología automática fiable. Esta es una línea de investigación que se debe continuar explorando para dar mayor solidez a los resultados y análisis que se realizan en trabajos como este.

Si bien este proyecto contó con evaluación humana, esta solo se centró en el método de derivaciones y sobre algunos de los ejemplos del dataset. Sería importante validar el resto de los resultados obtenidos con la evaluación automática mediante la realización de evaluación humana sobre el resto de los experimentos, en especial los experimentos de RAG que son los que muestran mayor competitividad con el método de derivaciones.

Otro aspecto a mejorar en la metodología de evaluación empleada, es la evaluación de la etapa de recuperación de información. En este proyecto se decidió

hacer una evaluación *end-to-end*, directamente sobre las respuestas generadas. Podría resultar beneficioso realizar una evaluación aislada del módulo de Retrieval para las diferentes técnicas exploradas (Bouza Heguerte, 2023).

Respecto a la definición de respuestas aceptables como aquellas con scores 3, 4 y 5 en la rúbrica definida para la evaluación con LLM, podría ser interesante tratar al score 3 como una categoría diferente. El análisis de esta categoría particular podría dejar más en evidencia las diferencias entre los métodos, y permitir una comparación más detallada.

Por último, el prototipo desplegado está siendo evaluado por los integrantes del EOC. La herramienta permite evaluar las respuestas generadas, por lo que se está obteniendo retroalimentación en casos de uso reales. Este *feedback* puede resultar valioso para pensar mejoras en el sistema implementado, así como poder validar la utilidad de este tipo de sistemas en un escenario real.

Referencias

- AI@Meta. (2024). Llama 3 model card. *GitHub*.
- Anthropic. (2024). The claude 3 model family: Opus, sonnet, haiku. -.
- Bahdanau, D., Cho, K., y Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. En Y. Bengio y Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*.
- Banerjee, S., y Lavie, A. (2005, junio). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. En *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (pp. 65–72). Ann Arbor, Michigan: Association for Computational Linguistics.
- Beltagy, I., Peters, M. E., y Cohan, A. (2020, diciembre). *Longformer: The Long-Document Transformer*. arXiv. (arXiv:2004.05150 [cs]) doi: 10.48550/arXiv.2004.05150
- Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., y Kolesnikov, A. (2022, June). Knowledge distillation: A good teacher is patient and consistent. En *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (cvpr)* (p. 10925-10934).
- Bouza Huguerte, L. (2023). *Recuperación de información para la búsqueda de respuestas en idioma español* (Tesis de maestría). Universidad de la República (Uruguay). Facultad de Ingeniería, Montevideo.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020). Language models are few-shot learners. En *Proceedings of the 34th international conference on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc.
- Bulian, J., Buck, C., Gajewski, W., Börschinger, B., y Schuster, T. (2022, diciembre). Tomayto, Tomahto. Beyond Token-level Answer Equivalence for Question Answering Evaluation. En *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (pp. 291–305). Abu Dhabi, United Arab Emirates: Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.20
- Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., y Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. En *Pml4dc at iclr 2020*.

- Choromanski, K. M., Likhoshesterov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., . . . Weller, A. (2020, octubre). Rethinking Attention with Performers..
- Cohen, J. (1960, abril). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46. doi: 10.1177/001316446002000104
- Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., y Salakhutdinov, R. (2019, julio). Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 2978–2988). Florence, Italy: Association for Computational Linguistics. doi: 10.18653/v1/P19-1285
- Dao, T. (2023, julio). *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*.
- Dao, T., Fu, D. Y., Ermon, S., Rudra, A., y Re, C. (2022, mayo). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness..
- Dettmers, T., Pagnoni, A., Holtzman, A., y Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. En A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, y S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 10088–10115). Curran Associates, Inc.
- Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019, junio). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (pp. 4171–4186). Minneapolis, Minnesota: Association for Computational Linguistics. doi: 10.18653/v1/N19-1423
- Fleiss, J. L. (1971, noviembre). Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5), 378–382. doi: 10.1037/h0031619
- Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., . . . Wang, H. (2024). *Retrieval-augmented generation for large language models: A survey*.
- Ghazarian, S., Weischedel, R., Galstyan, A., y Peng, N. (2019, noviembre). *Predictive Engagement: An Efficient Metric For Automatic Evaluation of Open-Domain Dialogue Systems*.
- Guu, K., Lee, K., Tung, Z., Pasupat, P., y Chang, M.-W. (2020). Realm: retrieval-augmented language model pre-training. En *Proceedings of the 37th international conference on machine learning*. JMLR.org.
- Hinton, G., Vinyals, O., y Dean, J. (2015). Distilling the knowledge in a neural network. En *Nips deep learning and representation learning workshop*.
- Hsieh, C.-Y., Li, C.-L., Yeh, C.-k., Nakhost, H., Fujii, Y., Ratner, A., . . . Pfister, T. (2023, julio). Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. En A. Rogers, J. Boyd-Graber, y N. Okazaki (Eds.), *Findings of the association for computational linguistics: Acl 2023* (pp. 8003–8017). Toronto, Canada: Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.507
- Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., . . . Chen, W. (2021, octubre). LoRA: Low-Rank Adaptation of Large Language Models..

- Huang, L., Ye, Z., Qin, J., Lin, L., y Liang, X. (2020, noviembre). GRADE: Automatic Graph-Enhanced Coherence Metric for Evaluating Open-Domain Dialogue Systems. En *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 9230–9240). Online: Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.742
- HuggingFace. (2021). *Question answering*. <https://huggingface.co/tasks/question-answering>. (Accessed: 2023-10-20)
- Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., ... Grave, E. (2024, mar). Atlas: few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24(1).
- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... Fung, P. (2023, mar). Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12). doi: 10.1145/3571730
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., ... Sayed, W. E. (2023). *Mistral 7b*.
- Jurafsky, D., y Martin, J. H. (2024). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models* (3.^a ed.). (Online manuscript released August 20, 2024)
- Kamalloo, E., Dziri, N., Clarke, C., y Rafiei, D. (2023, julio). Evaluating Open-Domain Question Answering in the Era of Large Language Models. En *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 5591–5606). Toronto, Canada: Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.307
- Kandpal, N., Deng, H., Roberts, A., Wallace, E., y Raffel, C. (2023). Large language models struggle to learn long-tail knowledge. En *Proceedings of the 40th international conference on machine learning*. JMLR.org.
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... Yih, W.-t. (2020, noviembre). Dense Passage Retrieval for Open-Domain Question Answering. En *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769–6781). Online: Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550
- Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., y Lewis, M. (2019, septiembre). Generalization through Memorization: Nearest Neighbor Language Models..
- Khattab, O., y Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over bert. En *Proceedings of the 43rd international acm sigir conference on research and development in information retrieval* (p. 39–48). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/3397271.3401075
- Kim, S., Shin, J., Cho, Y., Jang, J., Longpre, S., Lee, H., ... Seo, M. (2024). Prometheus: Inducing fine-grained evaluation capability in language models. En *The twelfth international conference on learning representations*.

- Kitaev, N., Kaiser, L., y Levskaia, A. (2019, septiembre). Reformer: The Efficient Transformer..
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., . . . Zettlemoyer, L. (2019, octubre). *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*.
- Lewis, P., Oguz, B., Rinott, R., Riedel, S., y Schwenk, H. (2020, julio). MLQA: Evaluating cross-lingual extractive question answering. En D. Jurafsky, J. Chai, N. Schluter, y J. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7315–7330). Online: Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.653
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., . . . Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. En *Proceedings of the 34th international conference on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc.
- Li, Z., Zhang, J., Fei, Z., Feng, Y., y Zhou, J. (2021, agosto). Conversations Are Not Flat: Modeling the Dynamic Information Flow across Dialogue Utterances. En *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 128–138). Online: Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.11
- Lin, C.-Y. (2004, julio). ROUGE: A Package for Automatic Evaluation of Summaries. En *Text Summarization Branches Out* (pp. 74–81). Barcelona, Spain: Association for Computational Linguistics.
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., y Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 157–173. doi: 10.1162/tacl.a_00638
- Lopez Latouche, G., Marcotte, L., y Swanson, B. (2023, julio). Generating Video Game Scripts with Style. En *Proceedings of the 5th Workshop on NLP for Conversational AI (NLP4ConvAI 2023)* (pp. 129–139). Toronto, Canada: Association for Computational Linguistics.
- Mehri, S., y Eskenazi, M. (2020a, julio). Unsupervised Evaluation of Interactive Dialog with DialoGPT. En *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 225–235). 1st virtual meeting: Association for Computational Linguistics.
- Mehri, S., y Eskenazi, M. (2020b, julio). USR: An Unsupervised and Reference Free Evaluation Metric for Dialog Generation. En *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 681–707). Online: Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.64
- Min, S., Boyd-Graber, J., Alberti, C., Chen, D., Choi, E., Collins, M., . . . Yih, W.-t. (2021, agosto). NeurIPS 2020 EfficientQA Competition: Systems,

- Analyses and Lessons Learned. En *Proceedings of the NeurIPS 2020 Competition and Demonstration Track* (pp. 86–111). PMLR. (ISSN: 2640-3498)
- Muennighoff, N., Tazi, N., Magne, L., y Reimers, N. (2022, octubre). *MTEB: Massive Text Embedding Benchmark*.
- Nogueira, R. F., y Cho, K. (2019). Passage re-ranking with BERT. *CoRR*, *abs/1901.04085*.
- Papineni, K., Roukos, S., Ward, T., y Zhu, W.-J. (2002, julio). Bleu: a Method for Automatic Evaluation of Machine Translation. En *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311–318). Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. doi: 10.3115/1073083.1073135
- Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P., Bakhtin, A., Wu, Y., y Miller, A. (2019, noviembre). Language Models as Knowledge Bases? En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 2463–2473). Hong Kong, China: Association for Computational Linguistics. doi: 10.18653/v1/D19-1250
- Phy, V., Zhao, Y., y Aizawa, A. (2020, diciembre). Deconstruct to Reconstruct a Configurable Evaluation Metric for Open-Domain Dialogue Systems. En *Proceedings of the 28th International Conference on Computational Linguistics* (pp. 4164–4178). Barcelona, Spain (Online): International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.368
- Rae, J. W., Potapenko, A., Jayakumar, S. M., Hillier, C., y Lillicrap, T. P. (2019, septiembre). Compressive Transformers for Long-Range Sequence Modelling..
- Rajpurkar, P., Zhang, J., Lopyrev, K., y Liang, P. (2016, noviembre). SQuAD: 100,000+ Questions for Machine Comprehension of Text. En *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (pp. 2383–2392). Austin, Texas: Association for Computational Linguistics. doi: 10.18653/v1/D16-1264
- Raschka, S. (2023, abril). *Parameter-Efficient LLM Finetuning With Low-Rank Adaptation (LoRA)*.
- Reimers, N., y Gurevych, I. (2019, noviembre). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. En K. Inui, J. Jiang, V. Ng, y X. Wan (Eds.), *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)* (pp. 3982–3992). Hong Kong, China: Association for Computational Linguistics. doi: 10.18653/v1/D19-1410
- Sai, A. B., Mohankumar, A. K., Arora, S., y Khapra, M. M. (2020, diciembre). Improving Dialog Evaluation with a Multi-reference Adversarial Dataset and Large Scale Pretraining. *Transactions of the Association for Computational Linguistics*, 8, 810–827. doi: 10.1162/tacl.a.00347
- Sastre, I., Moncecchi, G., y Rosá, A. (2024, November 13-15). Derivation promp-

- ting: A logic-based method for improving retrieval-augmented generation. En *Proceedings of the 18th ibero-american conference on artificial intelligence (iberamia 2024)*. Montevideo, Uruguay. (Accepted for publication)
- Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., . . . Yih, W.-t. (2024, junio). REPLUG: Retrieval-augmented black-box language models. En K. Duh, H. Gomez, y S. Bethard (Eds.), *Proceedings of the 2024 conference of the north american chapter of the association for computational linguistics: Human language technologies (volume 1: Long papers)* (pp. 8371–8384). Mexico City, Mexico: Association for Computational Linguistics.
- Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1), 72–101.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., . . . Lample, G. (2023, febrero). *LLaMA: Open and Efficient Foundation Language Models*. arXiv. (arXiv:2302.13971 [cs])
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., . . . Scialom, T. (2023, julio). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv. (arXiv:2307.09288 [cs]) doi: 10.48550/arXiv.2307.09288
- Valmeekam, K., Olmo, A., Sreedharan, S., y Kambhampati, S. (2022). Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). En *Neurips 2022 foundation models for decision making workshop*.
- Van Dalen, D. (2013). *Logic and Structure*. London: Springer. doi: 10.1007/978-1-4471-4558-5
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention is All you Need. En *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc.
- Vig, J. (2019, julio). A multiscale visualization of attention in the transformer model. En *Proceedings of the 57th annual meeting of the association for computational linguistics: System demonstrations* (pp. 37–42). Florence, Italy: Association for Computational Linguistics. doi: 10.18653/v1/P19-3007
- Wang, G., Cheng, S., Zhan, X., Li, X., Song, S., y Liu, Y. (2024). Openchat: Advancing open-source language models with mixed-quality data. En *The twelfth international conference on learning representations*.
- Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., . . . Wei, F. (2022, diciembre). *Text Embeddings by Weakly-Supervised Contrastive Pre-training*. arXiv. (arXiv:2212.03533 [cs]) doi: 10.48550/arXiv.2212.03533
- Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., y Wei, F. (2024). *Multilingual e5 text embeddings: A technical report*.
- Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., . . . Zhou, D. (2023). Self-consistency improves chain of thought reasoning in language models. En *The eleventh international conference on learning representations*.
- Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., . . . Le, Q. V. (2022). Finetuned language models are zero-shot learners. En *Internatio-*

- nal conference on learning representations.*
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., . . . Zhou, D. (2024). Chain-of-thought prompting elicits reasoning in large language models. En *Proceedings of the 36th international conference on neural information processing systems*. Red Hook, NY, USA: Curran Associates Inc.
- Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., y Nie, J.-Y. (2024). *C-pack: Packaged resources to advance general chinese embedding.*
- Xu, J., Fei, H., Pan, L., Liu, Q., Lee, M.-L., y Hsu, W. (2024). *Faithful logical reasoning via symbolic chain-of-thought.*
- Xu, P., Ping, W., Wu, X., McAfee, L., Zhu, C., Liu, Z., . . . Catanzaro, B. (2024). Retrieval meets long context large language models. En *The twelfth international conference on learning representations.*
- Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., y Narasimhan, K. R. (2023). Tree of thoughts: Deliberate problem solving with large language models. En *Thirty-seventh conference on neural information processing systems.*
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., y Cao, Y. (2023). React: Synergizing reasoning and acting in language models. En *The eleventh international conference on learning representations.*
- Yeh, Y.-T., Eskenazi, M., y Mehri, S. (2021, noviembre). A Comprehensive Assessment of Dialog Evaluation Metrics. En *The First Workshop on Evaluations and Assessments of Neural Conversation Systems* (pp. 15–33). Online: Association for Computational Linguistics. doi: 10.18653/v1/2021.eancs-1.3
- Yogatama, D., de Masson d’Autume, C., y Kong, L. (2021). Adaptive Semiparametric Language Models. *Transactions of the Association for Computational Linguistics*, 9, 362–373. (Place: Cambridge, MA Publisher: MIT Press) doi: 10.1162/tacl.a.00371
- Zelikman, E., Wu, Y., Mu, J., y Goodman, N. (2022). Star: Bootstrapping reasoning with reasoning. En S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, y A. Oh (Eds.), *Advances in neural information processing systems* (Vol. 35, pp. 15476–15488). Curran Associates, Inc.
- Zhang, C., Chen, Y., D’Haro, L. F., Zhang, Y., Friedrichs, T., Lee, G., y Li, H. (2021, agosto). DynaEval: Unifying Turn and Dialogue Level Evaluation. En *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (pp. 5676–5689). Online: Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.441
- Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., y Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. En *International conference on learning representations.*
- Zhao, X., Li, M., Lu, W., Weber, C., Lee, J. H., Chu, K., y Wermter, S. (2024, mayo). Enhancing zero-shot chain-of-thought reasoning in large language models through logic. En N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci,

- S. Sakti, y N. Xue (Eds.), *Proceedings of the 2024 joint international conference on computational linguistics, language resources and evaluation (lrec-coling 2024)* (pp. 6144–6166). Torino, Italia: ELRA and ICCL.
- Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., . . . Stoica, I. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. En A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, y S. Levine (Eds.), *Advances in neural information processing systems* (Vol. 36, pp. 46595–46623). Curran Associates, Inc.

Anexo A

Prompts

A continuación se presentan todas las prompts utilizadas en los experimentos realizados.

Los modelos que fueron ajustados con técnicas como instruction tuning y RLHF tienen un formato particular de prompt que sigue una estructura de tipo chat (con mensajes de usuario y asistente). Además, suelen incluir al principio un tipo de mensaje especial que contiene las instrucciones que el modelo de lenguaje debe seguir. A este mensaje se le llama “mensaje de sistema”.

Salvo que se aclare lo contrario, las prompts que se presentan obedecen esta abstracción en mensajes, y lo que se presenta es el mensaje de sistema. Además, si la prompt usa ejemplos few-shot, estos son agregados como mensajes de usuario y asistente.

Algunas de las prompts tienen secciones que están parametrizadas (por ejemplo, el contenido de las secciones con los *chunks* depende de cuales fueron recuperados). Las secciones parametrizadas de cada prompt se encuentran explicadas entre llaves.

A.1. Prompt utilizada en los ejemplos que no usan fuentes de información externas

Mensaje de sistema

Mantener una conversación sobre temas relacionados con la Facultad de

- ↪ Ingeniería de la Universidad de la República de Uruguay. Dar
- ↪ respuestas cortas, concretas y precisas. La conversación debe ser en
- ↪ español. Recuerda ser concreto.

Siempre agregar al final de la respuesta los links a las referencias que se

- ↪ utilizaron en la respuesta bajo el título de “**Referencias:**”.

A.2. Prompt utilizada en RAG

Mensaje de sistema

Mantener una conversación sobre extractos de documentos relacionados con la

- ↪ Facultad de Ingeniería de la Universidad de la República de Uruguay.
- ↪ Dar respuestas cortas, concretas y precisas. Utilizar únicamente la
- ↪ información de los siguientes extractos para construir la respuesta:

Extracto de página web de título: {Título del documento del primer chunk} –

- ↪ URL: {URL del documento del primer chunk}

{Texto del primer chunk}

...

Extracto de página web de título: {Título del documento del último chunk} –

- ↪ URL: {URL del documento del último chunk}

{Texto del último chunk}

Si los extractos no tienen una relación evidente con la pregunta, ignorarlos y

- ↪ responder que no se cuenta con información para responder la
- ↪ pregunta. Evitar las suposiciones. La conversación debe ser en español
- ↪ . Recuerda ser concreto.

Siempre agregar al final de la respuesta los links a las referencias que se

- ↪ utilizaron en la respuesta bajo el título de “**Referencias**”.

IMPORTANTE: Responder en español.

A.3. Prompts utilizadas en fine-tuning

A.3.1. Enfoque “Utilizar *chunks* necesarios”

Mensaje de sistema

A continuación se presentan tres extractos de documentos relacionados con la

- ↪ Facultad de Ingeniería de la Universidad de la República (UDELAR)
- ↪ de Uruguay:

--- EXTRACTO 1 ---

{Primer chunk}

---- EXTRACTO 2 ----

{Segundo chunk}

---- EXTRACTO 3 ----

{Tercer chunk}

---- FIN EXTRACTOS ----

Imagina que eres un estudiante de la facultad. Generar una pregunta que sea

- ↪ posible de responder a partir de algunos de los extractos anteriores.
- ↪ La pregunta puede ser tan creativa como lo desees, pero su respuesta
- ↪ se tiene que inferir de los extractos provistos.

Luego de escribir la pregunta, deberás escribir la respuesta y los extractos

- ↪ exactos utilizados para responder.

La pregunta debe estar escrita como lo haría un estudiante, no es necesario

- ↪ agregar demasiado contexto (por ejemplo, no es necesario mencionar a
- ↪ la facultad de ingeniería porque todas las preguntas están relacionadas
- ↪ a la facultad).

Respetar el siguiente formato (sin escribir una introducción ni nada extra):

****Pregunta:****

Escribir la pregunta generada directamente.

****Respuesta:****

Escribir la respuesta a la pregunta.

****Extractos usados:****

Listar los identificadores de los extractos necesarios para la respuesta

- ↪ separados por comas (por ejemplo, si se usa el extracto 1 poner: 1) (
- ↪ otro ejemplo, si se usan los extractos 1 y 3 poner: 1,3)

A.3.2. Enfoque “Utilizar todos los *chunks*”

Mensaje de sistema

A continuación se presentan extractos de documentos relacionados con la

- ↪ Facultad de Ingeniería de la Universidad de la República (UDELAR)
- ↪ de Uruguay:

---- EXTRACTO 1 ----

{Primer chunk}

...

---- EXTRACTO {n} ----

{Último chunk}

---- FIN EXTRACTOS ----

Imagina que eres un estudiante de la facultad. Generar una consulta que sea

- ↪ posible de responder usando TODOS los extractos anteriores. La
- ↪ consulta puede ser tan creativa como lo desees, pero su respuesta se
- ↪ tiene que inferir de los extractos provistos, usando al menos algo de
- ↪ información presente en cada extracto.

La consulta debe {formato de la pregunta elegido aleatoriamente}.

Luego de escribir la consulta, deberás escribir la respuesta.

Respetar el siguiente formato:

****Pregunta:****

Escribir la consulta generada directamente (sin escribir una introducción ni

- ↪ nada extra).

****Respuesta:****

Escribir la respuesta a la consulta.

A.3.3. Generación de preguntas fuera del dominio

Mensaje de sistema

Tu tarea es generar una consulta completamente aleatoria, sobre cualquier

- ↪ tema, para que un asistente te responda.

Ejemplos:

¿Dónde queda la pizzería Rodelú en Montevideo?

Tengo una emergencia!! Me duele la cabeza desde hace varios días, qué hago?!

¿Cómo se programa el Tetris?

A quién votarías en las próximas elecciones?

Quién eres?

Respetar el siguiente formato:

****Pregunta:****
Escribir una única consulta generada.

A.4. Prompts utilizadas en prompting de derivaciones

A.4.1. Prompt de derivación completa

Mensaje de sistema

Descripción de la tarea:
El objetivo es llegar a la respuesta de una pregunta, partiendo de un conjunto
→ de extractos que son información correcta (que llamaremos hipótesis).
→ Para ello, se debe aplicar reglas sobre las hipótesis que permiten
→ transformarlas y/o combinarlas para generar una conclusión, que a su
→ vez podría usarse como hipótesis, hasta llegar finalmente a la
→ respuesta esperada.

Reglas:

1. Extract:
Descripción: Dada una hipótesis compleja, esta regla genera una conclusión
→ que es una parte específica de la hipótesis.
Datos relevantes: – Solo puede tomar una (1) hipótesis. Si se quiere aplicar a
→ muchas hipótesis, se debe aplicar en varios pasos de forma secuencial.
→ – Solo puede generar una conclusión. Si se quiere generar múltiples
→ conclusiones se debe aplicar en varios pasos de forma secuencial.
Ejemplo: Si la hipótesis es “Hoy jugué al fútbol con unos amigos en la playa y
→ estuvo muy divertido”, la regla Extract podría generar la conclusión
→ “Hoy jugué al fútbol”.

2. Concat:
Descripción: Combina dos hipótesis independientes para generar una nueva
→ conclusión.
Datos relevantes: Debe tomar dos (2) o más hipótesis. – Solo puede generar
→ una conclusión. Si se quiere generar múltiples conclusiones se debe
→ aplicar en varios pasos de forma secuencial.
Ejemplo: Si tenemos las hipótesis “La deforestación afecta la biodiversidad” y
→ “El cambio climático es un problema global”, la regla Concat podría
→ generar la conclusión “La deforestación afecta la biodiversidad. Adem
→ ás, el cambio climático es un problema global”.

3. Instantiate:
Descripción: Genera una conclusión al instanciar una hipótesis genérica en un
→ caso particular.

Datos relevantes: Solo puede tomar una (1) hipótesis. Si se quiere aplicar a
→ muchas hipótesis, se debe aplicar en varios pasos de forma secuencial.
→ – Solo puede generar una conclusión. Si se quiere generar múltiples
→ conclusiones se debe aplicar en varios pasos de forma secuencial.
Ejemplo: Si la hipótesis genérica es “Los árboles son beneficiosos para el
→ medio ambiente”, Instantiate podría generar la conclusión “Los pinos
→ son beneficiosos para el medio ambiente”.

4. Compose:

Descripción: Combina dos hipótesis que comparten un elemento en común
→ para generar una nueva conclusión.
Datos relevantes: Solo puede tomar dos (2) hipótesis. Si se quiere aplicar a má
→ s de dos hipótesis, se debe aplicar en varios pasos de forma secuencial.
→ – Solo puede generar una conclusión. Si se quiere generar múltiples
→ conclusiones se debe aplicar en varios pasos de forma secuencial.
Ejemplo: Si tenemos las hipótesis “La deforestación afecta la biodiversidad” y
→ “La biodiversidad es esencial para la salud del planeta”, la regla
→ Compose podría generar la conclusión “La deforestación afecta la
→ salud del planeta”.

5. Refine:

Descripción: Adapta ligeramente la respuesta para que se ajuste mejor a la
→ pregunta, sin modificar la semántica ni el contenido de la hipótesis.
Datos relevantes: Solo puede tomar una (1) hipótesis. Si se quiere aplicar a
→ muchas hipótesis, se debe aplicar en varios pasos de forma secuencial.
→ – Solo puede generar una conclusión. Si se quiere generar múltiples
→ conclusiones se debe aplicar en varios pasos de forma secuencial.
Ejemplo: Si la respuesta es “Las abejas desempeñan un papel crucial en la
→ polinización”, la regla Refine podría adaptarla a “Las abejas desempe
→ ñan un papel crucial en la polinización de las flores”.

6. NoInfo:

Descripción: Se utiliza esta regla cuando ninguna de las hipótesis brinda
→ información para responder la pregunta (o parte de la pregunta).
Datos relevantes: No toma hipótesis. – Solo puede generar una conclusión. Si
→ se quiere generar múltiples conclusiones se debe aplicar en varios
→ pasos de forma secuencial.
Ejemplo: Si tenemos las hipótesis “La deforestación afecta la biodiversidad” y
→ “La biodiversidad es esencial para la salud del planeta”, pero la
→ pregunta es “¿Qué es la biodiversidad?”, no se cuenta con informació
→ n en las hipótesis para responder a la pregunta por lo que se debe
→ aplicar la regla NoInfo.
Importante: Como no lleva hipótesis, agregar como hipótesis: –1.

Ejemplo de Few-Shot

Lo que sigue es uno de los seis ejemplos Few-Shot contruidos manualmente.

Hipótesis:

{Se listan las hipótesis iniciales (chunks recuperados) del 1 al 3 (no se
↪ presentan por brevedad)}

Pregunta de usuario:

¿Cuál es el significado de un crédito y cuántas horas de estudio semanal se
↪ debe dedicar a una asignatura con 13 créditos?

Extract | 2 | Un crédito es una medida de la dedicación de una asignatura. Se
↪ supone que si una asignatura tiene más créditos, es necesario
↪ dedicarle más horas. | No es respuesta final

Nueva hipótesis:

a. Un crédito es una medida de la dedicación de una asignatura. Se supone
↪ que si una asignatura tiene más créditos, es necesario dedicarle más
↪ horas.

Extract | 2 | Un crédito equivale aproximadamente a una hora de estudio
↪ semanal, durante todo un semestre. | No es respuesta final

Nueva hipótesis:

b. Un crédito equivale aproximadamente a una hora de estudio semanal,
↪ durante todo un semestre.

Instantiate | b | Un crédito equivale aproximadamente a una hora de estudio
↪ semanal, durante todo un semestre. Por lo tanto, una asignatura con
↪ 13 créditos implica una dedicación aproximada de 13 horas semanales.
↪ | No es respuesta final

Nueva hipótesis:

c. Un crédito equivale aproximadamente a una hora de estudio semanal,
↪ durante todo un semestre. Por lo tanto, una asignatura con 13 cré
↪ ditos implica una dedicación aproximada de 13 horas semanales.

Concat | a,c | Un crédito es una medida de la dedicación de una asignatura.
↪ Se supone que si una asignatura tiene más créditos, es necesario
↪ dedicarle más horas. Un crédito equivale aproximadamente a una hora
↪ de estudio semanal, durante todo un semestre. Por lo tanto, una
↪ asignatura con 13 créditos implica una dedicación aproximada de 13
↪ horas semanales. | Es respuesta final

A.5. Prompt utilizada en grandes ventanas de contexto

Mensaje de sistema

```
Mantener una conversación sobre la Facultad de Ingeniería de la Universidad
↳ de la República de Uruguay. Dar respuestas cortas, concretas y
↳ precisas. Utilizar únicamente la información de los siguientes
↳ documentos para construir la respuesta:

<webpage>
Página web de título: {título del primer documento} – URL: {URL del
↳ primer documento}

{Texto del primer documento}
<\webpage>

...

<webpage>
Página web de título: {título del último documento} – URL: {URL del ú
↳ ltimo documento}

{Texto del último documento}
<\webpage>

Si los documentos no tienen una relación evidente con la pregunta, ignorarlos
↳ y responder que no se cuenta con información para responder la
↳ pregunta. Evitar las suposiciones. La conversación debe ser en español
↳ . Recuerda ser concreto.

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

A.6. Prompt utilizada para la limpieza de los mails

Esta prompt no sigue el formato de chat (con mensajes de sistema, usuario y asistente). En este caso, se envía la siguiente prompt directamente al modelo de lenguaje (notar que tiene dos ejemplos Few-Shot).

```
### INSTRUCCIONES:
```

Tu objetivo consiste en identificar y extraer la pregunta de un correo electrónico que incluye una consulta. Debes devolver únicamente la pregunta, eliminando cualquier otro contenido del correo que no sea la pregunta en sí. La pregunta extraída debe conservar todas las palabras utilizadas en el correo original para formularla. Además, es crucial incluir cualquier contexto necesario para comprender la pregunta de manera independiente y coherente. No responder nada más luego de la pregunta.

MAIL:

Buenas tardes, mi nombre es Juan Perez, cédula: 1.234567-8.

Quería consultar si para ingresar a la carrera de Licenciatura en Computación debo inscribirme a la carrera de Ingeniería en Computación o son independientes?

Muchas gracias, saludos,
Juan

PREGUNTA:

Para ingresar a la carrera de Licenciatura en Computación ¿Debo inscribirme a la carrera de Ingeniería en Computación o son independientes?

MAIL:

Buen día, mi hija esta haciendo 6° de economía. Puede inscribirse a ingeniería de producción? O tiene que dar algún examen aparte? Gracias

PREGUNTA:

Haciendo 6° de economía ¿Es posible inscribirse a ingeniería de producción o hay que dar algún examen aparte?

MAIL:

{Nuevo mail}

PREGUNTA:

A.7. Prompt utilizada para la evaluación

Mensaje de sistema

###Task Description:

An instruction (might include an Input inside it), a response to evaluate, a reference answer that gets a score of 5, and a score rubric representing a evaluation criteria are given.

1. Write a detailed feedback that assess the quality of the response strictly
↪ based on the given score rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5.
↪ You should refer to the score rubric.
3. The output format should look as follows: “Feedback: (write a feedback for
↪ criteria) [RESULT] (an integer number between 1 and 5)”
4. Please do not generate any other opening, closing, and explanations.

###The instruction to evaluate:
Responder la siguiente pregunta en español: {Pregunta}

###Response to evaluate:
{Respuesta generada a evaluar}

###Reference Answer (Score 5):
{Respuesta de referencia}

###Score Rubrics:

[Is the model response correct and truthful?]

Score 1: The response to evaluate contradicts the reference answer in a way
↪ that the information is untruthful in comparison to the reference
↪ answer.

Score 2: The response to evaluate has conflicts with the reference answer in a
↪ way that the information is partially untruthful in comparison to the
↪ reference answer.

Score 3: The response to evaluate has nothing to do with the reference answer
↪ , but it does not have conflicts with it. It does not contradict the
↪ reference answer.

Score 4: The response to evaluate partially matches semantically to the
↪ reference answer in a way that the information is truthful but not
↪ complete.

Score 5: The response to evaluate matches completely with the reference
↪ answer in a way that is truthful and has all the information that is
↪ expressed in the reference answer.

Anexo B

Documento de Estado del Arte (Octubre 2023)

B.1. Introducción

El presente trabajo condensa el estado del arte de la utilización de grandes modelos de lenguaje causales para la tarea de *generative question answering*, en el caso en que contamos con un conjunto de documentos D pertenecientes a un dominio específico sobre el cual se quiere responder preguntas. Antes de continuar, se definirán los conceptos previos necesarios para entender el resto del documento.

Un **modelo de lenguaje** es un modelo que asigna probabilidades a secuencias de palabras (Jurafsky y Martin, 2024). Se representará una secuencia w de n palabras como w_1, w_2, \dots, w_n , o de forma reducida como $w_{1:n}$. Para la probabilidad conjunta de cada palabra en una secuencia teniendo un valor particular $P(X_1 = w_1, X_2 = w_2, \dots, X_n = w_n)$ se utilizará la notación $P(w_1, w_2, \dots, w_n)$.

Utilizando la regla de la cadena de la probabilidad, se puede descomponer la probabilidad de la secuencia w de la siguiente manera:

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\dots P(w_n|w_{1:n-1})$$

Esta descomposición muestra que la tarea de modelado de lenguaje se puede resolver sabiendo calcular la probabilidad condicionada de una palabra dadas las palabras anteriores en la secuencia $P(w_n|w_{1:n-1})$.

Llamamos **grandes modelos de lenguaje** (Large Language Models - LLM) a modelos de aprendizaje profundo que utilizan la arquitectura Transformer (Vaswani y cols., 2017) y cuentan con cientos o miles de millones de parámetros. Algunos ejemplos conocidos son BERT (Devlin y cols., 2019) Llama (Touvron, Lavril, y cols., 2023), GPT-3 (Brown y cols., 2020), entre otros.

Existen dos tipos diferentes de grandes modelos de lenguaje: modelo de lenguaje causal y modelo de lenguaje enmascarado.

Un **modelo de lenguaje causal** es entrenado para la tarea de generar tokens de forma incremental mediante la acción repetida de predecir el siguiente token condicionado por los tokens anteriores, que pueden ser parte del contexto o tokens previamente predichos por el propio modelo (Jurafsky y Martin, 2024). GPT-3 y Llama son ejemplos de modelos de este tipo.

Por otro lado, un **modelo de lenguaje enmascarado** se entrena de forma bidireccional. Esto es, tomando como contexto tanto los tokens anteriores como posteriores de la secuencia. Luego, se entrena el modelo de lenguaje para predecir el token más probable que sustituya a un token particular (máscara) presente en la secuencia de tokens.

La tarea de **Question Answering (QA)** refiere a la construcción de sistemas capaces de responder preguntas en lenguaje natural de forma automática. La mayoría de sistemas de QA se centran en responder preguntas factoides, esto es, preguntas que pueden ser respondidas con hechos simples expresados en textos cortos (Jurafsky y Martin, 2024).

Existen dos variantes de sistemas de QA¹:

- **Extractive Question Answering:** El modelo debe extraer la respuesta de un contexto (por ejemplo, un conjunto de documentos). Esta variante se suele resolver utilizando modelos de lenguaje enmascarados, como BERT.
- **Generative Question Answering:** El modelo debe generar la respuesta como texto libre. Más precisamente, es posible clasificar estos sistemas en Open Generative QA si el modelo debe generar su respuesta a partir de un contexto, y Closed Generative QA si no se provee un contexto y la respuesta es completamente generada por el modelo. Para esta tarea, los modelos de lenguaje causales son más indicados por su naturaleza generativa.

A continuación, en la Sección B.2 se explorarán los trabajos relativos a modificar la arquitectura transformer para procesar segmentos de texto más grandes. En la Sección B.3 se comentará sobre técnicas de fine-tuning. En la Sección B.4 se estudiarán las técnicas que utilizan recuperación de información para seleccionar los documentos adecuados y modelos de lenguaje causal para generar la respuesta a partir de estos. En la Sección B.5 se estudiarán los métodos que utilizan modelos que permiten memorizar la información de manera explícita para complementar los modelos de lenguaje preentrenados. En la Sección B.6 se explorarán las técnicas de prompting relevantes a la tarea de generative question answering. Por último, en la Sección B.7 se realizará una exploración sobre los métodos de evaluación para esta tarea.

B.2. Ventana de contexto

Una primera solución natural al problema sería utilizar de manera directa el modelo de lenguaje causal para generar la respuesta a una entrada x , pasando

¹<https://huggingface.co/tasks/question-answering>

al modelo la concatenación de todos los documentos D y la entrada x como contexto: $P(y|D, x)$.

El problema inmediato que surge con esta solución es que los modelos de lenguaje basados en la arquitectura transformer tienen una ventana de contexto (largo máximo de la entrada) con un tamaño fijo y acotado. Por ejemplo, Llama 2 (Touvron, Martin, y cols., 2023) tiene una ventana de contexto de 4096 tokens, lo que equivale a aproximadamente 3000 palabras en inglés. Esto significa que si se tiene varios documentos o documentos extensos, es imposible ingresarlos todos en la entrada de un modelo de lenguaje con arquitectura transformer.

Esta limitación se debe a que la implementación estándar del modelo atencional tiene complejidad computacional $O(n^2d)$ y de memoria $O(n^2 + nd)$, donde n es el largo de la ventana de contexto y d es la dimensión de representación interna (Vaswani y cols., 2017). Esto limita el tamaño de la ventana de contexto a elegir al diseñar un modelo transformer. Múltiples trabajos buscan solventar esta limitación.

Transformer-XL (Dai y cols., 2019) es una arquitectura que introduce la noción de recurrencia a la arquitectura transformer. Se reutilizan los estados ocultos de los segmentos previos a la hora de computar el segmento actual, construyéndose una conexión recurrente entre segmentos. Para lograr esto, es también necesario calcular de forma diferente los *positional embeddings*, pasando de una representación absoluta a una relativa.

La diferencia entre la arquitectura transformer original y transformer-XL radica en que en la primera, tanto a la hora de entrenamiento como de inferencia, el modelo tiene en cuenta el segmento de texto de largo fijo que se está procesando en el momento, y no tiene información de los segmentos anteriores que fueron procesados.

En cambio, el modelo transformer-XL guarda el estado interno del segmento que está procesando en un *caché* para ser reutilizado en el siguiente segmento, como contexto extendido. Aunque no se modifica el gradiente en la etapa de entrenamiento, tener acceso al estado oculto del segmento anterior permite modelar dependencias de mayor alcance, más allá del tamaño de la ventana de contexto.

La figura B.1 muestra el manejo del contexto de la arquitectura transformer original, tanto en entrenamiento como en inferencia. La figura B.2, en cambio, muestra el funcionamiento de la arquitectura transformer-XL.

Rae, Potapenko, Jayakumar, Hillier, y Lillicrap (2019) toman la idea de transformer-XL y proponen la arquitectura Compressive Transformer que, en lugar de almacenar un número acotado de estados ocultos de segmentos anteriores en el *caché* y descartar los más antiguos, comprime estos estados ocultos sin descartarlos. Para esto se utiliza una nueva memoria llamada *compressed memory*, además de la memoria *caché* ya existente en el transformer-XL para almacenar los estados ocultos anteriores. Esta nueva memoria almacena el resultado de aplicar una función de compresión a los estados más antiguos que son descartados de la memoria *caché*.

Se proponen múltiples funciones de compresión, como *max/mean pooling*, *1D convolution*, *dilated convolutions* y *most used* (se preservan las más accedidas

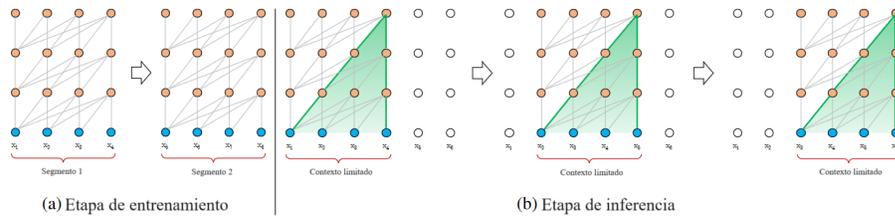


Figura B.1: Ilustración del modelo transformer original con ventana de contexto de tamaño 4, extraída y traducida de Dai y cols. (2019).

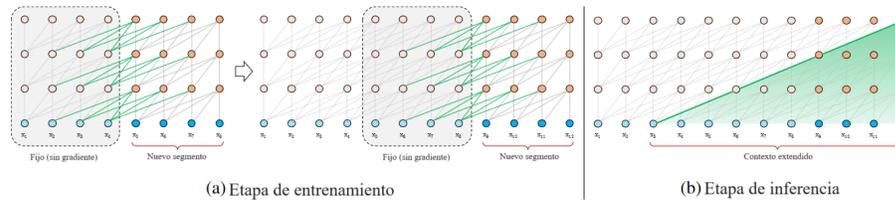


Figura B.2: Ilustración del modelo transformer-XL original con ventana de contexto de tamaño 4, extraída y traducida de Dai y cols. (2019).

por el modelo atencional).

Otros trabajos se centran en modificar el modelo atencional para mejorar su eficiencia computacional y de memoria. En la arquitectura *Reformer* (Kitaev y cols., 2019) se computa la atención de manera aproximada mediante el uso de *locality-sensitive hashing (LSH)*, que agrupa elementos similares en los mismos *buckets* con alta probabilidad. Así, solo se presta atención a elementos del mismo *bucket*. Esto permite cambiar el factor $O(n^2)$ de la complejidad de la atención por $O(n \log n)$.

Otra arquitectura propuesta es *Longformer* (Beltagy, Peters, y Cohan, 2020), en donde se utiliza un patrón de atención disperso basado en una ventana de tamaño fijo en torno a cada token. Se apilan múltiples capas de este tipo de atención resultando en un alto campo receptivo donde las capas superiores tienen la capacidad de incorporar información de toda la entrada en sus representaciones. Este método disminuye la complejidad a $O(nw)$, donde w es el tamaño de la ventana, lineal respecto al tamaño de la secuencia n .

Por otro lado, Choromanski y cols. (2020) proponen la arquitectura *Performer*, capaz de estimar de manera precisa la operación de atención pero disminuyendo el orden cuadrático a lineal. Para esto, proponen el mecanismo *Fast Attention Via positive Orthogonal Random features (FAVOR+)*.

Un enfoque distinto se presenta en Dao y cols. (2022), bajo el argumento de que los algoritmos propuestos hasta el momento no tienen en cuenta las lecturas y escrituras (*IO*) que se realizan en la memoria. La mayoría de las operaciones

de los transformers están limitadas por los accesos a la memoria y no por el cómputo.

Con esta motivación se propone *FlashAttention* (Dao y cols., 2022), un algoritmo que computa la atención de manera exacta con menos accesos a memoria. Dao (2023) propone luego *FlashAttention-2* que, bajo la misma motivación que el anterior, logra mejorar aproximadamente el doble en la velocidad de cómputo respecto a *FlashAttention*, acercándose al máximo rendimiento (*throughput*) alcanzable por la GPU.

El uso de *FlashAttention-2* permitió el desarrollo de *togethercomputer/LLaMA-2-7B-32K*², un modelo que incrementa a 32 mil tokens la ventana de contexto de Llama 2 (originalmente de 4096 tokens). Este modelo es resultado de continuar el pre-entrenamiento sobre el modelo Llama 2 con un corpus particular, y luego aplicar fine-tuning (véase sección B.3) centrándose en mejorar la capacidad de *few-shot* (véase sección B.6) en contextos largos.

B.3. Fine-tuning

Fine-tuning es el proceso de tomar la red de un modelo de lenguaje pre-entrenado y continuar el entrenamiento, muchas veces agregando una capa de clasificación que toma la capa superior del modelo como entrada, para realizar una tarea específica (*downstream task*) como puede ser *Named Entity Recognition* o *Question Answering*. La intuición es que la etapa de pre-entrenamiento sirve para aprender representaciones complejas de la semántica, y esto facilita al modelo a ser adaptado (fine-tuning) a los requerimientos de una tarea específica (Jurafsky y Martin, 2024).

Realizar fine-tuning puro sobre un LLM trae algunos inconvenientes que pueden ser difíciles de sobrellevar. En primer lugar, el resultado de esta tarea es un nuevo modelo con coeficientes diferentes al modelo original y por ende es necesario almacenarlo. Debido a la enorme cantidad de parámetros (la familia Llama 2 tiene modelos de entre 7 mil millones y 70 mil millones de parámetros), resulta inviable mantener nuevos modelos de esta envergadura cada vez que se realiza un fine-tuning. Por otro lado, realizar el fine-tuning es muy costoso computacionalmente y requiere mucho uso de memoria debido a la necesidad de ajustar todos los coeficientes del modelo pre-entrenado. Esta limitación hace que sea prohibitivo realizar fine-tuning sin usar hardware específico y costoso.

Para mitigar estas dificultades, se han propuesto distintos métodos que buscan mejorar la eficiencia del proceso de fine-tuning así como disminuir la cantidad de los parámetros a ajustar y almacenar.

Una técnica propuesta es *Low Rank Adaptation (LoRA)* (Hu y cols., 2021). Este método consiste en congelar los coeficientes del modelo pre-entrenado e inyectar matrices de descomposición de rango entrenables en cada capa de la arquitectura transformer. Esto permite reducir la cantidad de coeficientes entrenables, permitiendo disminuir los requerimientos de cómputo y de memoria así como de espacio de almacenamiento, ya que alcanza con almacenar solamente

²<https://huggingface.co/togethercomputer/LLaMA-2-7B-32K>



Figura B.3: Funcionamiento del modelo actualizado (fine-tuning) en etapa de inferencia utilizando *Low-Rank Adapters*. Figura traducida y adaptada de Raschka (2023).

las matrices de descomposición de rango, que poseen muchos menos parámetros que el modelo original.

Por ejemplo, supongamos que $W \in \mathbb{M}_{m \times n}$ es una matriz de pesos y $\Delta W \in \mathbb{M}_{m \times n}$ es la matriz de actualizaciones que queremos realizar a los pesos. En otras palabras, lo que se busca al realizar fine-tuning, en este ejemplo de una sola matriz de pesos, es aprender una matriz de pesos $W' \in \mathbb{M}_{m \times n}$ que resulta de modificar a la matriz de pesos pre-entrenada W . Esa diferencia resultante entre W y W' es representada por ΔW , por lo que se cumple que $W' = W + \Delta W$.

Como se muestra en Raschka (2023), en lugar de aprender ΔW , en *LoRA* se propone descomponer esta matriz en dos matrices de menor tamaño: $\Delta W = W_m W_n$, donde $W_m \in \mathbb{M}_{m \times r}$, $W_n \in \mathbb{M}_{r \times n}$ y r es un número pequeño llamado rango. Manteniendo los pesos originales congelados y solamente entrenando estas nuevas matrices permite reducir la cantidad de coeficientes entrenables de $m * n$ a $m * r + r * n$. Si r es un valor pequeño, la diferencia puede ser muy considerable. Luego del entrenamiento, podemos reconstruir la matriz de pesos del modelo con fine-tuning W' sumando a los pesos originales W el producto de las dos matrices aprendidas: $W' = W + W_m W_n$. La figura B.3 muestra el funcionamiento del modelo con fine-tuning en etapa de inferencia.

En *QLoRA* (Dettmers y cols., 2023) se toma esta idea y se aplica sobre modelos cuantizados. Cuantizar un modelo significa representar los coeficientes y activaciones del modelo utilizando tipos de datos de poca precisión como enteros de 8 bits (int8) o de 4 bits (int4) en lugar de usar punto flotante de 32 bits (float32)³. Esta reducción permite requerir una menor cantidad de memoria a la hora de utilizar el modelo en etapa de inferencia.

Sin embargo, no era posible realizar fine-tuning sobre los modelos ya cuantizados sin degradar el rendimiento. *QLoRA* es una técnica de cuantización a 4

³https://huggingface.co/docs/optimum/concept_guides/quantization

bits que permite agregar un conjunto de matrices *Low Rank Adapter (LoRA)* que son ajustadas mediante backpropagation a través de los pesos ya cuantizados, sin degradación de rendimiento.

Esto reduce los requerimientos de memoria para realizar fine-tuning notoriamente. Por ejemplo, realizar fine-tuning sobre un modelo de 65 mil millones de parámetros requeriría más de 780GB de memoria de GPU. Mediante la técnica *QLoRA*, se reduce a menos de 48GB de memoria.

Independientemente del método de fine-tuning que se utilice, un cuestionamiento razonable a hacerse es si efectivamente se está mejorando el rendimiento del LLM en la tarea objetivo, en comparación con utilizar técnicas de prompting como *few-shot* (véase sección B.6). En Wei y cols. (2022) se propone realizar fine-tuning con datasets adaptados de una manera particular, a lo que se le llama *instruction tuning*. Se toma más de 60 datasets de tareas de PLN y se adaptan para ser expresados mediante instrucciones en lenguaje natural.

La idea es que al enseñar al modelo a realizar tareas definidas mediante instrucciones, este será capaz de aprender a seguir instrucciones incluso para tareas no vistas en la etapa de fine-tuning. Para evaluar este comportamiento, se agrupan los datasets en *clusters* del mismo tipo de tarea, y se evalúa con cada uno de estos realizando *instruction tuning* con el resto de los *clusters*.

Mediante *intruction tuning*, en Wei y cols. (2022) se entrena el modelo FLAN a partir del modelo pre-entrenado LAMBDA-PT. El trabajo concluye que FLAN supera a LAMBDA-PT en todas las tareas utilizando *zero-shot* e incluso supera a GPT-3 (Brown y cols., 2020) con *few-shot* en 10 datasets, demostrando la efectividad de realizar *instruction tuning*.

B.4. Retrieval Augmented Generation

Si bien se ha observado que los modelos de lenguaje preentrenados adquieren bastante información y conocimiento a partir de los datos de entrenamiento sin acceder a información externa (Petroni y cols., 2019), existen múltiples desventajas en el uso de estos modelos. Por ejemplo, no es posible actualizar el conocimiento en tiempo real ni obtener fuentes a partir de las cuales se obtiene la información. Además, pueden generar alucinaciones (información que no se desprende de los datos de entrenamiento, inventada).

Para mitigar estos problemas, se propone el método de Retrieval Augmented Generation (RAG) (P. Lewis, Perez, y cols., 2020). Este método continúa la tendencia de combinar memoria paramétrica (i.e. modelos de lenguaje pre-entrenados) con memoria no paramétrica (mediante técnicas de recuperación de información), como también se propone en el método REALM (Guu y cols., 2020), en donde se utilizan modelos de lenguaje enmascarados.

El método RAG consiste en dos componentes preentrenados:

- **Retriever** $p_{\eta}(z|x)$, con parámetros η utilizado para obtener los k documentos z que están más relacionados con la entrada x .

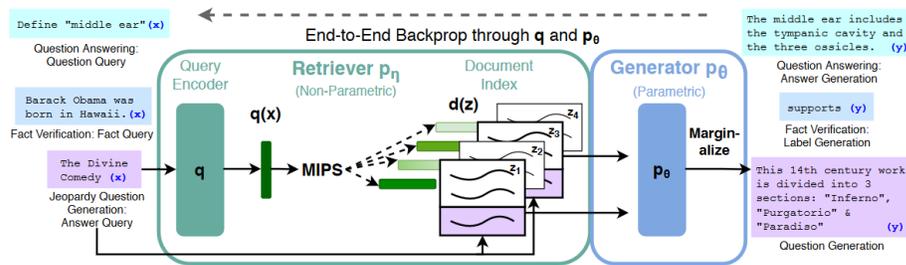


Figura B.4: Método RAG. El retriever es un modelo DPR (véase sección B.4.1) y el generator es un modelo de lenguaje. Obtenida de (P. Lewis, Perez, y cols., 2020).

- **Generator** $p_\theta(y_i|x, z, y_{1:i-1})$, con parámetros θ que cumple la tarea de modelo de lenguaje causal, basándose en los previos tokens generados $y_{1:i-1}$, la entrada x y el documento recuperado z .

La generación utilizando este método se puede realizar de dos maneras diferentes:

- **RAG-Token:** se recuperan k documentos nuevos z_i para cada token que se va generando. De esta manera, el generator es influenciado por diferentes documentos a medida que va produciendo la respuesta. Para obtener la distribución de probabilidad para el siguiente token y_i , se calcula el promedio de $p_\theta(y_i|x, z_i, y_{1:i-1})$ para cada uno de los k documentos z_i ponderado por $p_\eta(z_i|x)$.
- **RAG-Sequence:** se recuperan los k documentos z_i utilizando el retriever y se genera la secuencia completa utilizando esos documentos como contexto. De esta manera, se generan las secuencias completas y utilizando el generator con cada documento z_i como contexto. Luego es posible obtener probabilidades para cada una de las secuencias generadas tomando el promedio de $p_\theta(y|x, z_i)$, ponderando por $p_\eta(z_i|x)$.

La figura B.4 muestra los dos componentes mencionados. En P. Lewis, Perez, y cols. (2020), estos dos componentes luego son combinados y entrenados end-to-end mediante la técnica de fine-tuning. Sin embargo, también es posible combinar ambos modelos utilizando técnicas de prompting sin necesidad de pasar por una etapa de entrenamiento.

B.4.1. Retriever

El retriever cumple el rol del modelo no paramétrico, utilizado para la tarea de recuperación de información. Si bien en P. Lewis, Perez, y cols. (2020) se utilizan representaciones densas, obtenidas mediante modelos paramétricos,

es posible utilizar cualquier método de recuperación de información para el retriever (teniendo en cuenta que no siempre es posible realizar un fine-tuning end-to-end).

En [P. Lewis, Perez, y cols. \(2020\)](#) se utiliza Dense Passage Retriever (DPR) ([Karpukhin y cols., 2020](#)) como modelo de retriever. DPR utiliza una arquitectura bi-encoder. Primero, se usa un encoder $E_P(\cdot)$ que asocia cada documento z con un vector $v_z \in \mathbb{R}^d$. Al momento de procesar una pregunta q , se utiliza un encoder diferente $E_Q(\cdot)$ que asocia la pregunta con un vector $v_q \in \mathbb{R}^d$. Finalmente se obtienen los k documentos más cercanos según las distancias (o similitud) entre los vectores v_z asociados a los documentos y el vector v_q asociado a la pregunta. Se utiliza el producto punto para calcular la similitud entre vectores.

Se han propuesto otros métodos de representaciones densas que podrían ser utilizados para este componente. En particular, existen varios trabajos que proponen modelos para la generación de sentence embeddings (embeddings de textos de largo arbitrario). A diferencia de DPR, estos modelos utilizan un solo encoder para generar los embeddings tanto de los documentos como de la pregunta o entrada. Se utiliza la similitud del coseno entre los embeddings de los documentos y de la pregunta para recuperar los k documentos más similares.

Un método comúnmente utilizado para la generación de sentence embeddings es usar la salida del primer token (CLS), o el promedio de la capa de salida, de un modelo de lenguaje enmascarado (por ejemplo, BERT ([Devlin y cols., 2019](#))). El modelo Sentence-BERT ([Reimers y Gurevych, 2019](#)) mejora los resultados de estos embeddings modificando la arquitectura de BERT para incorporar la estructura de redes siamesas en la etapa de entrenamiento, como se observa en la figura [B.5](#).

En [L. Wang y cols. \(2022\)](#) se presenta E5 (EmbEddings from bidirEctional Encoder rEpresentations), con el objetivo de generar embeddings de propósito general. Para ello se construye un dataset a partir de datos sin etiquetar, lo que permite entrenar con una mayor cantidad de datos que en los métodos previamente mencionados.

Debido a la gran cantidad de modelos disponibles para generación de sentence embeddings, es necesario una manera de evaluarlos para las diferentes tareas que pueden ser utilizados. Por ello se construyó MTEB (Massive Text Embedding Benchmark) ([Muennighoff y cols., 2022](#)) que cubre 8 tareas en las que se utilizan embeddings, incluida la tarea de retriever. Actualmente se mantiene una tabla de clasificación utilizando dicho benchmark, accesible en [hugging face](https://huggingface.co/spaces/mteb/leaderboard)⁴.

B.4.2. Generator

El generator puede ser modelado usando cualquier modelo de lenguaje causal, esto es, que predice el siguiente token en una secuencia de tokens. En [P. Lewis, Perez, y cols. \(2020\)](#) se usa BART-large ([M. Lewis y cols., 2019](#)), un modelo transformer seq2seq preentrenado. Se concatena la entrada x con el documento obtenido z para utilizar como contexto de entrada del modelo.

⁴<https://huggingface.co/spaces/mteb/leaderboard>

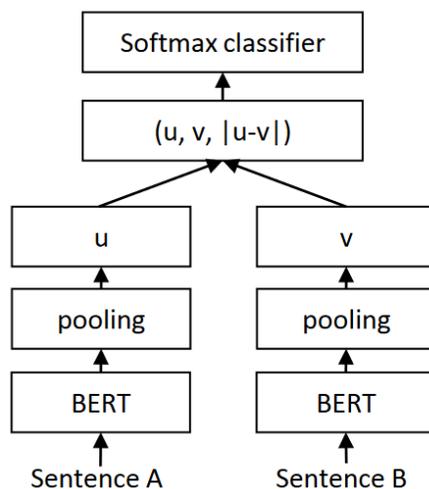


Figura B.5: Arquitectura de sentence-BERT. Las dos redes BERT tienen los mismos pesos (redes siamesas). Obtenida de (Reimers y Gurevych, 2019).

B.4.3. Ventana de contexto y retrieval augmentation

En P. Xu y cols. (2024) se proponen investigar la efectividad de realizar *Retrieval Augmentation* en modelos con diferentes tamaños de ventana de contexto. Se busca responder si modelos con que utilizan esta técnica tienen resultados competitivos con modelos que usan grandes ventanas de contexto, y si combinar ambas cosas tiene mejor rendimiento que cada una por separado.

Para llevar a cabo los experimentos, utilizan un modelo propietario de 43 mil millones de parámetros y Llama 2 (Touvron, Martin, y cols., 2023) en su versión de 70 mil millones de parámetros, aumentando las ventanas de contexto de estos modelos de 4k a 16k, y en el caso de Llama 2 también a 32k.

Experimentan con tres técnicas de retrievers diferentes y los extractos seleccionados con k -Nearest Neighbors son concatenados y utilizados directamente como entrada al modelo de lenguaje.

Los experimentos muestran que los modelos con ventana de 4k y sin retrieval obtienen los peores resultados. Esto es evidente porque el tamaño de la ventana no alcanza para ingresar todo el contexto y por ende se recorta de forma aleatoria. Se observa que utilizar *Retrieval Augmentation* ayuda considerablemente a los modelos con ventanas de 4k.

Otra observación interesante es que aún utilizando *Retrieval-Augmentation* con $k = 5$ como parámetro de k -Nearest Neighbors (en otras palabras, utilizando el mismo contexto como entrada para los distintos modelos), los modelos con ventanas de contexto más grandes obtienen mejores resultados. Esto se podría explicar por el fenómeno de “*lost in the middle*” (Liu y cols., 2024): los modelos tienden a prestar más atención a los tokens que se encuentran al principio y al

final de la entrada, y menor atención a los tokens en el medio.

Se concluye que *Retrieval Augmentation* mejora de forma significativa el rendimiento de ambos modelos independientemente del tamaño de la ventana de contexto. Los mejores resultados se obtienen con el modelo de Llama 2 con tamaño de ventana de 32k y *Retrieval Augmentation*, por lo que el uso del retriever es relevante aún con ventanas de contexto grandes. Además, los modelos con ventanas de 4k y *Retrieval Augmentation* son competitivos con los modelos de 16k que no utilizan esta técnica.

B.5. Modelos de memorización explícita

Otro método explorado en la literatura para solventar parte de los problemas presentes en los modelos de lenguajes preentrenados, consiste en utilizar modelos de lenguaje con memorización explícita para complementar a los modelos preentrenados, que solamente cuentan con el conocimiento implícito en sus parámetros. Para esto, se realiza una interpolación lineal entre las distribuciones de probabilidad de un modelo de lenguaje preentrenado y un modelo con memorización explícita (Khandelwal y cols., 2019).

Se introduce el método kNN-LM (*k*-Nearest Neighbors Language Modeling) (Khandelwal y cols., 2019), en el cual, a partir de un corpus de texto D (que podría ser un conjunto de documentos), se construye una colección de tipo clave-valor donde los valores son los tokens. Dicha colección es accedida en tiempo de inferencia y cumple el rol del modelo de lenguaje con memorización explícita.

B.5.1. Construcción de la colección

Para la construcción de la colección, se cuenta con una función $f(\cdot)$ que asigna una representación vectorial de largo fijo a un contexto (secuencia de tokens) $c_i = (w_1, w_2, \dots, w_{i-1})$. Esta función puede ser implementada con cualquiera de los métodos ya mencionados en la sección B.4.1.

Luego, cada documento d del conjunto de datos D es convertido a un conjunto de pares de la forma (c_i, w_i) recorriendo la secuencia de tokens de izquierda a derecha y definiendo el contexto para cada token como todos los tokens anteriores a este en el documento d .

Finalmente, la colección $(\mathcal{K}, \mathcal{V})$ es construida aplicando la función $f(\cdot)$ a cada contexto c_i , obteniendo un vector de largo fijo. Este vector cumple el rol de clave, y w_i el rol de valor:

$$(\mathcal{K}, \mathcal{V}) = \{(f(c_i), w_i) | (c_i, w_i) \in D\}$$

B.5.2. Inferencia

Luego de construida la colección $(\mathcal{K}, \mathcal{V})$, es posible definir una distribución $P_{\text{kNN}}(w|x)$ sobre todos los tokens posibles dada la entrada x siguiendo los

siguientes pasos:

1. Se obtienen k entradas de la colección aplicando k -Nearest Neighbors, tomando la representación vectorial $f(x)$ de la entrada y comparando con las claves de la colección utilizando una función de distancia $d(\cdot, \cdot)$.
2. Se computa la distribución sobre los tokens obtenidos aplicando *Softmax* a las distancias negadas, y sumando las probabilidades para los tokens con más de una ocurrencia en los k recuperados. Los tokens que no fueron obtenidos tienen probabilidad 0.

Por otro lado, se cuenta con la distribución $P_{\text{LM}}(w|x)$ que se obtiene a partir de un modelo de lenguaje preentrenado.

Entonces, se obtiene la distribución final mediante la interpolación de ambas distribuciones $P_{\text{kNN}}(w|x)$ y $P_{\text{LM}}(w|x)$, usando un parámetro λ :

$$P(w|x) = \lambda P_{\text{kNN}}(w|x) + (1 - \lambda) P_{\text{LM}}(w|x)$$

B.5.3. Elección del parámetro λ

Es necesario elegir un valor para el parámetro de interpolación λ . Distintos trabajos proponen diferentes maneras de tomar ese valor, algunos de forma estática y otros de manera dinámica al momento de generar el siguiente token.

[Khandelwal y cols. \(2019\)](#) ajustan el parámetro con el conjunto de validación, y luego se utiliza de forma estática. Esto trae como desventaja que siempre se utiliza el mismo valor de λ , independientemente del contexto y las distribuciones de probabilidad P_{kNN} y P_{LM} .

Para solventar este problema, [Lopez Latouche, Marcotte, y Swanson \(2023\)](#) proponen aprender el parámetro λ a partir de la concatenación del último estado oculto del transformer, las distancias entre $f(x)$ y las claves de los k elementos obtenidos con k -Nearest Neighbors, y las probabilidades de estos k tokens según la distribución $P_{\text{LM}}(w|x)$. Se denota a este nuevo vector como h_i . A partir de este vector, se calcula el valor de λ aplicando la función sigmoid al producto de una matriz de pesos W con h_i :

$$\lambda = \sigma(Wh_i)$$

[Yogatama, de Masson d'Autume, y Kong \(2021\)](#) exploran una manera diferente de combinar los tokens obtenidos de la colección con la distribución del modelo preentrenado. Utilizan una arquitectura similar a kNN-LM, tomando la arquitectura Transformer-XL ([Dai y cols., 2019](#)) para ampliar la memoria a corto plazo del modelo de lenguaje no paramétrico y una colección de tipo clave-valor para la memoria a largo plazo. Al igual que [Khandelwal y cols. \(2019\)](#), se realiza k -Nearest Neighbors para obtener las entradas más similares. La principal diferencia consiste en utilizar un mecanismo de compuerta (*gating*) para combinar la información obtenida con la colección clave-valor y el modelo de lenguaje preentrenado.

Para esto, en primer lugar se obtienen los word embeddings (utilizados en el transformer) correspondientes a cada uno de los k tokens recuperados al realizar k -Nearest Neighbors, denotados y_1, y_2, \dots, y_k . Luego, se implementa el mecanismo de compuerta como se describe a continuación.

Primero se hace una agregación de la información de y_1, y_2, \dots, y_k con un mecanismo de atención (Bahdanau y cols., 2015) usando el estado oculto h correspondiente al último token de la última capa del transformer como el *attention query*:

$$m = \sum_{i=1}^k \frac{\exp y_i^\top h}{\sum_{j=1}^k \exp y_j^\top h} y_i$$

Luego, se calcula la compuerta g , dependiente del contexto, que decide cuánto se necesita la información local (h) versus la información de largo plazo (m). Al igual que en Lopez Latouche y cols. (2023), se debe aprender una matriz de pesos V y se aplica sigmoid al producto de la matriz con el estado oculto h :

$$g = \sigma(Vh)$$

Por último, se calculan los valores para la nueva distribución:

$$z = (1 - g) \odot m + g \odot h$$

$$P(w|x) = \text{softmax}(z)$$

B.6. Técnicas de prompting

El trabajo que presentó al modelo GPT-3 (Brown y cols., 2020) vino acompañado de un concepto novedoso emergente del uso de los LLMs llamado *In-Context Learning*.

In-Context Learning refiere a usar el texto de entrada de un modelo de lenguaje pre-entrenado para especificar una tarea. El modelo es condicionado por una instrucción en lenguaje natural y/o algunas demostraciones de la tarea (ejemplos) y se espera que complete nuevas instancias de la tarea simplemente prediciendo la continuación a dicha entrada, sin modificar de ninguna forma los pesos del modelo.

Se discriminan distintos tipos de *In-Context learning*:

- *Few-Shot* hace referencia a el escenario en donde el modelo recibe un conjunto chico de ejemplos de una tarea en tiempo de inferencia para condicionar su salida. Opcionalmente se cuenta con una descripción de la tarea al principio de la entrada.
- *One-Shot* hace referencia al caso particular de *Few-Shot* en donde solo se utiliza un ejemplo.
- *Zero-Shot* consiste únicamente de la descripción de la tarea en lenguaje natural, sin utilizar ningún ejemplo adicional.



Figura B.6: Ejemplo de *Zero-Shot*, *One-Shot* y *Few-Shot* aplicado a la tarea de traducción de inglés a francés. Figura traducida de [Brown y cols. \(2020\)](#).

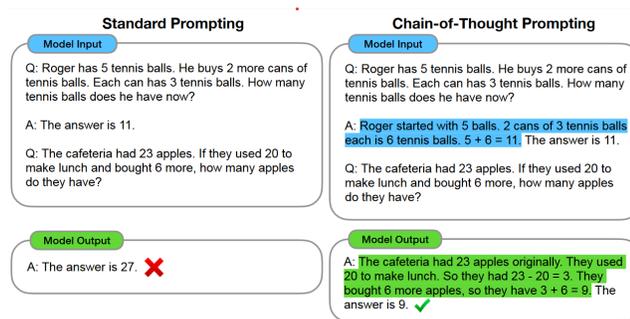


Figura B.7: Ejemplo de un problema aritmético resuelto con *One-Shot* de forma incorrecta y resuelto con *Chain-of-Thought* de forma correcta. Figura obtenida de [Wei y cols. \(2024\)](#).

La figura B.6 muestra un ejemplo aplicando los diferentes tipos de *In-Context learning* a la tarea de traducción.

A partir de este trabajo y a medida que los LLMs fueron más accesibles, comenzaron a surgir varias técnicas novedosas para mejorar el rendimiento de estos modelos en tareas específicas solamente mediante la utilización del texto de entrada que recibe el modelo. A este texto se le llamará *prompt*.

Una técnica de prompting que mejora el rendimiento de *Few-Shot* es *Chain-of-Thought* ([Wei y cols., 2024](#)). El objetivo de esta técnica es que el modelo genere una cadena de razonamiento, esto es, una serie coherente de pasos de razonamiento intermedios que conducen a la respuesta final de un problema.

Para lograr esto, se aplica *Few-Shot prompting*, y se agrega a cada ejemplo una cadena de razonamiento hasta el resultado esperado. La figura B.7 muestra un ejemplo de *Chain-of-Thought prompting*.

El trabajo muestra que LLMs suficientemente grandes son capaces de generar cadenas de razonamiento si se presentan ejemplos siguiendo la técnica *Few-Shot*. Además, se presentan resultados para experimentos empíricos sobre razonamiento aritmético, razonamiento de sentido común y razonamiento simbólico.

Para el razonamiento aritmético, se concluye que *Chain-of-Thought* mejora los resultados respecto a *Few-Shot* pero solo con modelos que superan aproximadamente los 100 mil millones de parámetros. Sin embargo, en los escenarios de razonamiento de sentido común y razonamiento simbólico, se observan mejoras para casi todos los *benchmarks* en todos los tamaños de modelos que se probaron.

X. Wang y cols. (2023) continúan el trabajo anterior introduciendo *Self Consistency prompting*. Esta técnica consiste en ejecutar *Chain-of-Thought prompting* múltiples veces sobre la misma entrada utilizando *sampling decoding*⁵, para generar un conjunto diverso de cadenas de razonamiento. Se elige como respuesta final el resultado que más veces se repite en todas las generaciones.

Se evalúa *Self-Consistency* en tareas de razonamiento aritmético, de sentido común y simbólico utilizando varios modelos de diferentes tamaños y se concluye que mejora a *Chain-of-Thought* con un margen alto en todas las tareas evaluadas.

Otra técnica que surge inspirada en *Chain-of-Thought* es *Tree-of-Thoughts* (Yao, Yu, y cols., 2023), un paradigma que permite a los modelos explorar varios caminos de razonamiento distintos. Para lograr esto, se interpreta al razonamiento como una búsqueda sobre un árbol. La motivación para este enfoque está en que métodos como *Chain-of-Thought* no permiten explorar diferentes continuaciones en una misma cadena de razonamiento, ni hacer *backtracking* cuando se llega a conclusiones equivocadas.

La figura B.8 compara de forma esquemática las técnicas de *Zero-Shot*, *Chain-of-Thought*, *Self-Consistency* y *Tree-of-Thoughts*.

Se diseñan los siguientes pasos para seguir el enfoque de *Tree-of-Thoughts*:

- **Descomposición en pasos de razonamiento:** A diferencia de *Chain-of-Thought*, es necesario dividir la cadena de razonamiento en pasos, donde cada paso corresponderá a un nodo del árbol. Un paso de razonamiento puede ser una oración, una ecuación, un párrafo, etcétera.
- **Generador de pasos de razonamiento:** Se deben generar k candidatos a pasos de razonamiento, a partir de un nodo padre, utilizando un LLM. Se presentan dos estrategias para generarlos: (1) Usar *Chain-of-Thought prompting* para generar los k candidatos utilizando una estrategia de *sampling decoding*, (2) generar los k candidatos utilizando una misma prompt, lo que permite evitar duplicados.
- **Evaluador de estado:** Dados los k candidatos a paso de razonamiento, el evaluador de estado sirve para evaluar el progreso obtenido con cada candidato en pos de resolver el problema. Es necesario para determinar cuales estados seguir explorando y cuales no. Se proponen dos estrategias de evaluación utilizando LLMs: (1) Asignar un valor a cada candidato

⁵ *Sampling* es un método de decodificación (método para elegir el siguiente token dada la distribución de probabilidad sobre todos los tokens) que consiste en tomar el token aleatoriamente según la distribución de probabilidad $P(w_n|w_{1:n-1})$. Existen distintas variantes de *sampling*, como por ejemplo *random sampling*, *top-k sampling* y *nucleus sampling*.

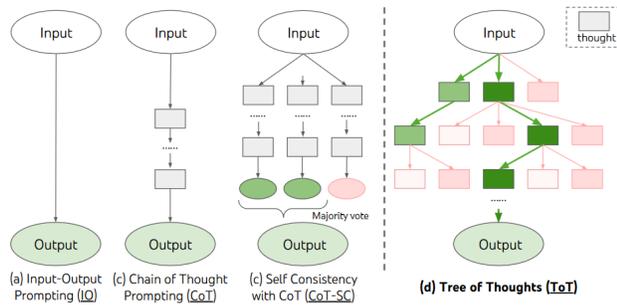


Figura B.8: Ilustración esquemática donde cada rectángulo representa un paso de razonamiento. Se compara *Zero-Shot (IO)*, *Chain-of-Thought (CoT)*, *Self-Consistency (SC-CoT)* y *Tree-of-Thoughts (ToT)*. Figura obtenida de (Yao, Yu, y cols., 2023).

de forma independiente. El valor podría ser una etiqueta (por ejemplo: *sure/likely/impossible*) o un número (por ejemplo: 1-10). (2) Votar sobre los posibles candidatos utilizando una prompt que permita comparar todos los candidatos y elegir el más adecuado (o definir un orden).

- **Algoritmo de búsqueda:** Se debe elegir un algoritmo de búsqueda para recorrer el árbol. En el trabajo se exploran los algoritmos *Breath-first search* y *Depth-first search*.

Se evalúa esta técnica para resolver el juego *Game of 24*, para escritura creativa y para resolver crucigramas 5x5. En estos escenarios se observa que *Tree-of-Thoughts* obtiene mejores resultados en distintas métricas que *Chain-of-Thought* y *Few-Shot*.

Otro trabajo que utiliza la idea de razonamiento es *React* (Yao, Zhao, y cols., 2023), en donde se busca integrar el razonamiento con la posibilidad de realizar acciones.

La técnica *React* busca generar pasos de razonamiento y acciones de manera intercalada para resolver una tarea particular. Esto permite generar planes mediante razonamientos para ejecutar acciones (*reason to act*), así como interactuar con fuentes externas (por ejemplo Wikipedia) para incorporar información adicional a los razonamientos (*act to reason*).

Se genera una prompt utilizando *Few-Shot* para generar acciones (dentro de un conjunto predefinido) y razonamientos en lenguaje natural. Cada ejemplo utilizado en la prompt consiste en una secuencia de acciones, razonamientos y observaciones del ambiente, generada por un humano para resolver una instancia de un problema.

Se pretende que el modelo genere los pasos de razonamiento y las acciones. Estas últimas generan una observación del ambiente que es agregada a la prompt para que el modelo continúe la secuencia.

Los experimentos llevados a cabo incluyen utilizar la técnica mediante *Few-Shot prompting* así como realizar fine-tuning de modelos más pequeños para que aprendan el formato de las secuencias. Además, se experimenta con combinar *React* y *Self-Consistency (CoT-SC)* de dos maneras diferentes: (a) **React** \rightarrow **CoT-SC**: Cuando *React* no llega a un resultado en una cantidad fija de pasos, pasar a *CoT-SC*. (b) **CoT-SC** \rightarrow **React**: Cuando el resultado que ocurre mayoritariamente en las n generaciones de *CoT-SC* ocurre menos de $n/2$ veces, pasar a *React*.

Esta técnica de prompting es evaluada empíricamente en tareas de *question answering*, *fact verification*, el juego *ALFWorld* basado en textos y navegación de sitios web. Se concluye que la combinación de *React* y *CoT-SC* da los mejores resultados respecto a realizar solamente acciones o *CoT-SC*. Sin embargo, se observa que con *React* se obtienen mejores resultados utilizando fine-tuning en lugar de *Few-Shot prompting*.

B.7. Métricas de evaluación

Originalmente, la manera de evaluar sistemas diseñados para la tarea de *Question Answering* consistía en utilizar métricas de *lexical matching* entre respuestas de un *gold standard* (provistas en un *benchmark*) y las predicciones del modelo. Sin embargo, a medida que se empieza a utilizar modelos generativos en lugar de extractivos, los métodos clásicos de evaluación empiezan a fallar. Esto sucede debido a que la lista de respuestas del *gold standard* es incompleta y no es posible cubrirlas todas. Además, los LLMs pueden generar respuestas con alucinaciones que parecen plausibles pero son incorrectas, haciendo aún más difícil la comparación (Kamalloo y cols., 2023).

En Kamalloo y cols. (2023), se evalúan 12 modelos de *open-domain question answering* con distintas estrategias para medir las discrepancias entre el rendimiento medido y el rendimiento real de estos modelos.

Una estrategia realizada es evaluar utilizando las métricas más comúnmente utilizadas de *lexical matching* que se definen en Rajpurkar y cols. (2016) de la siguiente manera:

- **Exact-Match accuracy (EM)**: La respuesta candidata es considerada correcta si se encuentra completamente en el conjunto de respuestas del *gold standard*. *EM accuracy* se calcula como el ratio de respuestas correctas en el conjunto de test.
- **F1 score**: Se calcula el F1 en función del solapamiento de los tokens de la respuesta candidata con los tokens de cada respuesta del *gold standard* (sin importar el orden). Se toma el F1 más alto entre el conjunto de respuestas del *gold standard* para cada respuesta candidata, y se calcula el promedio sobre todos los valores de F1.

Otra estrategia de evaluación investigada en Kamalloo y cols. (2023) es mediante similitud semántica, considerando la tarea de evaluación como un problema de clasificación donde el objetivo es decidir si la respuesta candidata y

la respuesta de referencia son semánticamente similares o no. Se utiliza el modelo BEM (Bulian y cols., 2022), entrenado siguiendo la arquitectura de BERT con un conjunto de datos especialmente construido para la tarea de *Answer Equivalence*.

También se utiliza como métrica de evaluación *Zero-Shot prompting* con un LLM. Se construye una prompt que contiene la pregunta, la respuesta candidata y la respuesta de referencia. Finalmente, se agrega la instrucción de determinar si la respuesta candidata es correcta.

Por último, se realiza evaluación humana para comparar todas estas estrategias de evaluación con el criterio humano. Algunos trabajos recurren directamente a evaluación humana para evaluar sistemas de QA. En Min y cols. (2021) se observa que el rendimiento de estos sistemas aumenta un 23% en promedio al ser evaluados por humanos, en comparación con la evaluación automática.

Luego de evaluar los 12 sistemas con estas estrategias, se observa que la correlación τ de Kendall entre la evaluación humana y la evaluación usando *Zero-Shot prompting* y BEM es de 0.75 y 0.70 respectivamente, mientras que EM y F1 tienen una correlación de 0.23 y 0.37 respectivamente.

Se observa que los sistemas con LLMs son severamente subestimados por las métricas de *lexical matching*. De la evaluación humana se concluye que estos sistemas son los que producen los mejores resultados, y además son los que obtienen el mayor aumento respecto a las evaluaciones obtenidas con EM y F1.

De todas formas, se observa que los sistemas que usan LLMs tienden a ser evaluados de forma menos precisa por las estrategias *Zero-Shot* y BEM cuando son demasiado largas y tienen información inventada.

Otro trabajo que hace una valoración de distintas métricas es (Yeh y cols., 2021), centrándose en la evaluación para sistemas de diálogo, una tarea con muchos puntos en común con *question answering*. Se evalúan 23 métricas de evaluación diferentes en 10 datasets distintos.

Se clasifican estas métricas en función de tres aspectos: (1) Si usa un modelo pre-entrenado, (2) qué dataset se utiliza para entrenarla, y (3) si requiere una respuesta de referencia o no.

Se observa que las métricas basadas en reglas (BLEU (Papineni y cols., 2002), ROUGE (Lin, 2004) y METEOR (Banerjee y Lavie, 2005)) tienen problemas para evaluar respuestas generadas por los sistemas de diálogo que usan LLMs, debido a que se requiere un entendimiento semántico más profundo del contexto del diálogo y la respuesta generada.

Se concluye que USR (Mehri y Eskenazi, 2020b), GRADE (Huang y cols., 2020), DEB (Sai y cols., 2020), y USL-H (Phy y cols., 2020) son las métricas con mejor rendimiento para evaluar la generación de respuestas (*turn-level evaluation*).

Por otro lado, FlowScore (Li y cols., 2021), PredictiveEngage (Ghazarian y cols., 2019), DynaEval (C. Zhang y cols., 2021) y FED (Mehri y Eskenazi, 2020a) dan mejores resultados para evaluar el diálogo completo (*dialog-level evaluation*).

También se observa que las métricas tienen un mejor rendimiento evaluando sobre los conjuntos de datos que fueron entrenados. Por lo tanto, se concluye

que es importante buscar métodos para adaptar las métricas existentes a nuevos dominios.