

Desarrollo de un Asistente Inteligente para la atención al cliente y solución técnica en empresas del sector del agua mediante modelos preentrenados

Profesor Colaborador: Jonathan Ferrer Mestres.
Profesor Responsable: Susana Acebo Nadal.

UOC

- **Estudios:**
Grado de Ingeniería Informática
- **Asignatura:**
Trabajo Final de Grado
- **Curso:**
2023/24 (marzo 2024/junio 2024)
- **Nombre Estudiante:**
Antonio Fernández Salcedo
- **Área del trabajo final:**
Inteligencia Artificial – Memoria

Fecha de entrega:
16 de junio 2024





Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada.
[3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Desarrollo de un Asistente Inteligente para la atención al cliente y solución técnica en empresas del sector del agua mediante modelos preentrenados</i>
Nombre del autor:	<i>Antonio Fernández Salcedo</i>
Nombre del consultor/a:	<i>Jonathan Ferrer Mestres</i>
Nombre del PRA:	<i>Susana Acebo Nadal</i>
Fecha de entrega (mm/aaaa):	06/2024
Titulación:	<i>Grado de Ingeniería Informática</i>
Área del Trabajo Final:	<i>Inteligencia Artificial</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>BERT, Mixtral, Fine-Tuning.</i>
Resumen del Trabajo	
<p>El proyecto aborda el desarrollo y evaluación de un asistente inteligente basado en modelos preentrenados de lenguaje natural, con el objetivo de crear un sistema capaz de ofrecer respuestas precisas y mejorar la atención a los usuarios del Canal de Isabel II mediante técnicas de inteligencia artificial y NLP.</p> <p>La metodología sigue un enfoque en cascada, con la recolección progresiva y el análisis continuo de un conjunto de datos de preguntas y respuestas etiquetadas del Canal de Isabel II. Se selecciona, ajusta y entrena el modelo BERT mediante técnicas de <i>fine-tuning</i>. Además, se implementa un mecanismo de filtrado basado en el <i>score</i> de confianza para mejorar la calidad de las respuestas generadas.</p> <p>Para realizar las inferencias se elige el modelo Mixtral debido a su capacidad para seguir instrucciones específicas y guiar a los clientes o técnicos del Canal de Isabel II en la resolución de problemas o la realización de tareas. En el diseño del interfaz se utiliza el <i>framework</i> Hugging Face.</p>	

En la evaluación del rendimiento de los modelos se utilizan diversas métricas léxicas. También, se analiza y compara la capacidad de diferentes modelos de LLM para proporcionar respuestas a consultas específicas de los usuarios de la empresa Canal de Isabel II.

En conclusión, el proyecto sienta las bases para un asistente inteligente efectivo en la atención a usuarios del Canal de Isabel II. Además, se proponen trabajos futuros con ajustes y mejoras adicionales para alcanzar su máximo potencial en términos de precisión y calidad de las respuestas.

Abstract

The project addresses the development and evaluation of an intelligent assistant based on pre-trained natural language models. Its goal is to create a system capable of providing accurate answers and improving user support for Canal de Isabel II through artificial intelligence and NLP techniques.

The methodology follows a cascading approach, involving progressive data collection and continuous analysis of labeled question-and-answer datasets from Canal de Isabel II. The BERT model is selected, fine-tuned, and trained using specific techniques. Additionally, a confidence-based filtering mechanism is implemented to enhance the quality of generated responses.

For making inferences, the Mixtral model is chosen due to its ability to follow specific instructions and guide Canal de Isabel II customers or technicians in problem-solving and task execution. The interface design utilizes the Hugging Face framework.

Performance evaluation of the models employs various lexical metrics. Furthermore, the capacity of different LLM models to provide responses to specific queries from Canal de Isabel II users is analyzed and compared.

In conclusion, the project lays the groundwork for an effective intelligent assistant in user support for Canal de Isabel II. Besides, future work with additional adjustments and improvements is proposed to reach its full potential in terms of accuracy and quality of responses.

Índice

1. Introducción	9
1.1 Contexto y justificación del Trabajo	9
1.2 Objetivos del Trabajo.....	10
1.3 Enfoque y método seguido	11
1.4 Planificación del Trabajo	16
1.5 Breve resumen de productos obtenidos	21
1.6 Breve descripción de los otros capítulos de la memoria	22
2. Investigar la aplicabilidad de modelos preentrenados	24
2.1. Revisar la literatura existente sobre el uso de modelos preentrenados.....	24
2.2. Análisis de modelos preentrenados para fine-tuning.....	28
2.3. Recopilación y análisis del conjunto de datos.....	33
3. Realizar fine-tuning e inferencia en modelos preentrenados para adaptarlos al contexto del Canal de Isabel II	35
3.1. Seleccionar modelos preentrenados adecuados para fine-tuning.....	35
3.2. Definir los parámetros y características específicas del fine-tuning para adaptar el modelo al contexto del Canal de Isabel II.....	38
3.3. Seleccionar un modelo preentrenado adecuado para ajustar parámetros de generación texto o realizar tareas de procesamiento del lenguaje natural (PLN) para una inferencia específica en el contexto Canal de Isabel II.....	47
4. Desarrollar un asistente que guíe a los clientes y/u ofrezca soluciones técnicas especializadas	51
4.1. Implementar el asistente inteligente con un modelo preentrenado ajustado.....	51
4.2. Diseñar la interfaz del asistente para facilitar la interacción con los clientes o técnicos del Canal de Isabel II.....	58
4.3. Incorporar capacidades de ofrecer soluciones técnicas especializadas a consultas específicas del sector del agua.....	65

5. Evaluación del asistente	67
5.1. Recopilación de datos para la evaluación del asistente.....	67
5.2. Valorar la precisión de las respuestas.....	69
5.3. Evaluar y comparar las repuestas con otros LLM.....	71
5.4. Identificar áreas de mejora.....	79
6. Resultados y productos obtenidos	80
7. Conclusiones	83
8. Glosario	86
9. Bibliografía	89
10. Anexos	93
10.1. Anexo I: Diagrama Gantt: Tareas del Proyecto.....	93
10.2. Anexo II: Tabla de hitos finales.....	95
10.3. Anexo III: BERT en español.....	96
10.4. Anexo IV: Mixtral: Comprensión y funcionamiento.....	99
10.5. Anexo V: Aceleradores por Hardware en Google Colab.....	107
10.6. Anexo VI: Inferencia de LLMs.....	115
10.7. Anexo VII: Fine-tuning a grandes modelos de lenguaje.....	118
10.8. Anexo VIII: PEFT (Ajuste fino eficiente en parámetros).....	121
10.9. Anexo IX: LoRA y QLoRA: Cuantización de modelos de lenguaje.....	124
10.10. Anexo X: Respuestas obtenidas.....	128
10.11. Anexo XI: Comparativa con LLM.....	131
10.12. Anexo XII: Diagrama de Despliegue.....	137
10.13. Anexo XIII: Estructuras del modelo BERT	138
10.14. Anexo XIV: Código del proyecto.....	140

Lista de figuras

- Imagen 1: Analyzing Semantic Equivalence of Sentences Using BERT.....26
- Imagen 2: Esquema de modelo de respuesta a preguntas28
- Imagen 3: Contexto de prueba, común para el análisis de los modelos.....28
- Imagen 4: Código de prueba de modelos en Colab.....29
- Imagen 5: Código de prueba de modelos en Colab.....30
- Imagen 6: Código de prueba de modelos en Colab.....31
- Imagen 7: Código de prueba de modelos en Colab.....32
- Imagen 8: Data Set en formato JSON.....34
- Imagen 9: Argumentos que configuran el entrenamiento del modelo BERT38
- Imagen 10: Archivo config.json.....41
- Imagen 11: Archivo special_tokens_map.json.....44
- Imagen 12: Modelo BERT
<https://huggingface.co/spaces/Antonio49/BERTfinetuningContextoPredefinido2>.....51
- Imagen 13: Tarjeta modelo. <https://huggingface.co/Antonio49/Personal>.....55
- Imagen 14: Archivos tarjeta. <https://huggingface.co/Antonio49/Personal/tree/main> 56
- Imagen 15: Modelo transformers. <https://huggingface.co/Antonio49/Personal>.....57
- Imagen 16: Modelo Mixtral usuarios Canal Isabel II.....58
- Imagen 17: Función de generación de respuestas del modelo Mixtral.....62
- Imagen 18: Modelo Mixtral experto en sector del agua.....66
- Imagen 19: Archivos del espacio del modelo Mixtral.....66
- Imagen 20: Modelos de HuggingChat. <https://huggingface.co/chat/models>.....71
- Imagen 21: Configuración de LLM con especificaciones para inferencias.72
- Imagen 22: Asistentes^{BETA}, LLM populares.....73
- Imagen 23: Espacios creados en Hugging Face. <https://huggingface.co/Antonio49>.82
- Imagen 24: Diagrama Gantt. Tareas del Proyecto.....93
- Imagen 25: Diagrama Gantt. Tareas del Proyecto.94

- Imagen 26: Router selecciona los expertos #1 y #6 de la arquitectura Mixtral.....99
- Imagen 27: Estructura del modelo Mixtral-8x7B.....104
- Imagen 28: Fine_tune_Mixtral_8x7B_(QLoRA). Ejecución.....106
- Imagen 29: Comparativa de hardware en aceleradores de Google Colab.....110
- Imagen 30: Tipo de tiempo de ejecución en Google Colab.....113
- Imagen 31: Instancia de GPU, T4 GPU.....114
- Imagen 32: Ajuste fino completo de los modelos sin enfoque PEFT.....123
- Imagen 33: Ajuste de modelo de lenguaje grande en la GPU.....127
- Imagen 34: BERT_fine-tuning. Preguntas y Respuestas del Canal de Isabel II.....128
- Imagen 35: Asistente de usuarios/clientes de Canal de Isabel II.....129
- Imagen 36: Experto en sector del agua del Canal de Isabel II.....130
- Imagen 37: Inferencia al modelo LLM google/gemma-1.1-7b-it.....131
- Imagen 38: Inferencia al modelo LLM mistralai/mistral-7B-Instruct-v0.2.....132
- Imagen 39: Inferencia al modelo LLM mistralai/Mixtral-8x7B-Instruct-v0.1.....133
- Imagen 40: Inferencia al modelo LLM CohereForAI/c4ai-command-r-plus.....134
- Imagen 41: Inferencia al modelo LLM NousResearch/Mixtral-8x7B-DPO.....135
- Imagen 42: Inferencia al modelo LLM Mixtral-8x7B en el espacio creado.....136
- Imagen 43: Diagrama de despliegue en Hugging Face.....137
- Imagen 44: Estructura del model BERT inicial.....138
- Imagen 45: Estructura del model BERT con capas agregadas.....139

1. Introducción

1.1 Contexto y justificación del Trabajo.

En la actualidad, el sector del agua enfrenta desafíos significativos, incluyendo la creciente complejidad de las consultas de los clientes y la necesidad de proporcionar soluciones técnicas eficientes. La demanda de servicios eficaces en este ámbito requiere una respuesta adaptativa y tecnológica que pueda abordar de manera precisa y oportuna las consultas variadas y las problemáticas técnicas específicas.

La complejidad de las operaciones en empresas del sector del agua, como el Canal de Isabel II, ha llevado a la identificación de la necesidad de un asistente inteligente. Este asistente debe ser capaz de no solo comprender las consultas de los clientes, sino también ofrecer soluciones técnicas especializadas de manera eficiente. La implementación de inteligencia artificial en forma de un asistente es esencial para mejorar la atención al cliente y la resolución de problemas técnicos.

Este proyecto se enfocará en la implementación de modelos preentrenados de lenguaje, como BERT o Mixtral, para llevar a cabo inferencia y/o fine-tuning y desarrollar un asistente inteligente. La temática es relevante en el contexto actual, ya que la aplicación de inteligencia artificial en la mejora de servicios y la toma de decisiones es fundamental para la eficiencia operativa y la satisfacción del cliente.

1.2 Objetivos del Trabajo.

A. Objetivo general:

El objetivo principal de este TFG es el desarrollo de un asistente inteligente diseñado específicamente para empresas del sector del agua, con un enfoque particular en el Canal de Isabel II. Este asistente busca mejorar la atención al cliente y ofrecer soluciones técnicas especializadas mediante la aplicación de modelos preentrenados.

B. Objetivos específicos:

1. *Investigar la aplicabilidad de modelos preentrenados:*
 - a. Revisar la literatura existente sobre el uso de modelos preentrenados, como BERT o Mixtral, en aplicaciones similares.
 - b. Identificar casos de éxito y desafíos en la implementación de modelos preentrenados.
2. *Realizar inferencia y/o fine-tuning en un modelo preentrenado para adaptarlo al contexto del Canal de Isabel II:*
 - c. Seleccionar un modelo preentrenado adecuado para inferencia y/o fine-tuning.
 - d. Definir los parámetros y características específicas de la inferencia y/o el fine-tuning para adaptar el modelo al contexto del Canal de Isabel II.
3. *Desarrollar un asistente que guíe a los clientes y ofrezca soluciones técnicas especializadas:*
 - e. Implementar el asistente inteligente utilizando el modelo preentrenado ajustado.
 - f. Diseñar la interfaz del asistente para facilitar la interacción con los clientes y técnicos del Canal de Isabel II.
 - g. Incorporar capacidades de ofrecer soluciones técnicas especializadas a consultas específicas del sector del agua.

1.3 Enfoque y método seguido

A. Enfoque en cascada

En este proyecto se seguirá un enfoque de desarrollo en cascada, caracterizado por una secuencia lineal y rígida de fases bien definidas. Este enfoque se considera adecuado para este proyecto debido a la necesidad de un desarrollo estructurado y controlado, asegurando la calidad y la completitud del asistente inteligente.

1. Ventajas del enfoque en cascada:

- Claridad y simplicidad: Las fases del método son claras y bien definidas, facilitando la comprensión y gestión del proyecto.
- Control y predicción: Cada fase del proyecto tiene un objetivo definido y un tiempo estimado, permitiendo un control y predicción del desarrollo.
- Estabilidad: El enfoque en cascada promueve la estabilidad del proyecto, ya que los cambios en los requisitos son menos probables después de la finalización de cada fase.

2. Desventajas del enfoque en cascada:

- Inflexibilidad: Una vez que se inicia una fase, es difícil realizar cambios significativos en el diseño o los requisitos.
- Falta de adaptación: El enfoque en cascada no se adapta bien a cambios en los requisitos o a la necesidad de prototipado rápido.
- Riesgo de errores: Si se cometen errores en las primeras fases, pueden ser costosos y difíciles de corregir en las fases posteriores.

3. Consideraciones adicionales:

A pesar de las limitaciones del enfoque en cascada, se implementarán estrategias para mitigar sus desventajas:

- Prototipado: Se utilizarán técnicas de prototipado para validar el diseño y las funcionalidades del asistente antes de la implementación completa.
- Pruebas tempranas: Se realizarán pruebas en las primeras fases del desarrollo para detectar errores de forma temprana y evitar costos mayores en las etapas finales.

En conclusión, el enfoque en cascada se considera adecuado para el desarrollo del asistente inteligente, ya que proporciona una estructura clara y controlada. Sin embargo, se implementarán estrategias para mitigar las limitaciones del enfoque y asegurar la flexibilidad y adaptabilidad del proyecto a lo largo de su desarrollo.

B. Estrategias posibles

En el desarrollo de este asistente inteligente para empresas del sector del agua:

1. Desarrollo de un producto nuevo:

Ventajas:

- Control total sobre la arquitectura y funcionalidades del asistente.
- Mayor flexibilidad para adaptar el asistente a las necesidades específicas del Canal de Isabel II.
- Potencial para la creación de un producto innovador y diferenciador en el mercado.

Desventajas:

- Mayor tiempo y recursos necesarios para el desarrollo.
- Riesgo de no alcanzar los objetivos deseados debido a la complejidad del proyecto.
- Mayor dificultad para la integración con sistemas existentes.

2. Adaptación de un producto existente:

Ventajas:

- Menor tiempo y recursos necesarios para el desarrollo.
- Posibilidad de aprovechar funcionalidades y experiencia previas de otros proyectos.
- Mayor facilidad para la integración con sistemas existentes.

Desventajas:

- Menor control sobre la arquitectura y funcionalidades del asistente.
- Posible limitación para adaptar el asistente a las necesidades específicas del Canal de Isabel II.
- Riesgo de no alcanzar el nivel de rendimiento y precisión deseado.

3. Estrategia elegida: Adaptación de un modelo preentrenado:

Se opta por la adaptación de un modelo preentrenado como la estrategia más adecuada para el desarrollo del asistente inteligente. Esta decisión se basa en las siguientes razones:

- Ahorro de tiempo de desarrollo y recursos: La adaptación de un modelo preentrenado permite aprovechar el trabajo y conocimiento ya invertidos en el desarrollo de modelos de lenguaje natural de última generación. De esta forma, se reduce significativamente el tiempo y los recursos necesarios para llevar a cabo el proyecto.
- Aprovechamiento de funcionalidades previas: Los modelos preentrenados, como BERT o Mixtral, ofrecen una amplia gama de funcionalidades avanzadas, como la comprensión del lenguaje natural, la generación de texto y la traducción automática. Adaptar estos modelos permite aprovechar estas funcionalidades sin necesidad de desarrollarlas desde cero.
- Flexibilidad para la adaptación y la mejora: A pesar de la elección de un modelo preentrenado, la estrategia propuesta permite un alto grado de flexibilidad para adaptarlo a las necesidades específicas del Canal de Isabel II. El proceso de inferencia y/o fine-tuning y la evaluación continua permiten ajustar el modelo a las características del dominio del sector del agua y mejorar el rendimiento del asistente inteligente a lo largo del tiempo, asegurando un rendimiento óptimo para la tarea de atención al cliente.
- Integración con sistemas existentes: La adaptación de un modelo preentrenado facilita la integración con los sistemas informáticos existentes en el Canal de Isabel II. Esto permite una rápida implementación del asistente inteligente en el entorno operativo de la empresa.
- Reducción del riesgo: La elección de una estrategia basada en la adaptación de un modelo preentrenado reduce el riesgo de no alcanzar los objetivos deseados. Los modelos preentrenados han demostrado un alto nivel de rendimiento en diversas tareas, lo que proporciona una base sólida para el desarrollo del asistente inteligente.

En resumen, la estrategia de adaptar un modelo preentrenado se considera la más adecuada para el desarrollo del asistente inteligente, ya que permite un equilibrio óptimo entre la eficiencia, la flexibilidad y el rendimiento.

C. Estrategia de desarrollo

En este proyecto se seguirá una estrategia de desarrollo basada en la adaptación de varios modelos preentrenados: Mixtral y BERT. Esta estrategia combinará las fortalezas de ambos modelos para crear un asistente inteligente robusto y efectivo para el sector del agua.

1. Selección de modelos preentrenados

- **Mixtral**: Este modelo ha demostrado un alto rendimiento en tareas de comprensión del lenguaje natural en español, lo que lo convierte en una opción atractiva para la atención al cliente en el sector del agua.
- **BERT**: Este modelo de Google AI es reconocido por su capacidad para procesar y comprender información textual compleja, siendo una herramienta valiosa para la extracción de información y la generación de respuestas precisas.

2. Fine-tuning

Se aplicará la técnica de fine-tuning para adaptar los modelos preentrenados al dominio específico del sector del agua. El proceso de fine-tuning consistirá en:

- **Recopilación de un conjunto de datos de preguntas y respuestas del Canal de Isabel II**: Este conjunto de datos servirá como base para entrenar los modelos y ajustarlos a las necesidades específicas de la empresa.
- **Preprocesamiento del conjunto de datos**: Se aplicarán técnicas de limpieza y preparación de datos para asegurar la calidad y consistencia de la información.
- **Ajuste de los hiperparámetros de los modelos**: Se optimizarán los parámetros de los modelos para obtener el mejor rendimiento posible en las tareas de atención al cliente.

3. Inferencia

Una vez finalizado el proceso de fine-tuning, los modelos se integrarán en un sistema de inferencia que permita su uso en tiempo real. El sistema de inferencia se encargará de:

- Procesar las consultas de los clientes: Las consultas se recibirán en formato textual y se procesarán para extraer la información relevante.
- Seleccionar el modelo más adecuado: En función de los resultados obtenidos, se seleccionará el modelo preentrenado (Mixtral o BERT) que mejor se adapte a la tarea.
- Generar una respuesta: El modelo seleccionado generará una respuesta precisa y personalizada a la consulta del cliente o técnico.

4. Evaluación

Se implementarán métricas de evaluación para medir el rendimiento del asistente inteligente en diferentes escenarios. Estas métricas permitirán:

- Valorar la precisión de las respuestas: Se analizará la exactitud de las respuestas generadas por el asistente en comparación con la información real.
- Evaluar y comparar la repuesta con otros LLM: Se realizarán pruebas con otros modelos como ChatGPT o Copilot y evaluar una comparativa en función del comportamiento y de la precisión de las respuestas.
- Identificar áreas de mejora: A partir de la evaluación, se identificarán áreas en las que se puede mejorar el rendimiento del asistente.

5. Despliegue

Se publicarán los asistentes en la nube para que cualquier usuario pueda realizar consultas de forma automática.

- Preparación de los asistentes para el despliegue.
- Publicación en una plataforma.
- Pruebas de despliegue: Verificar el correcto funcionamiento del asistente en la plataforma.

En definitiva, la estrategia de adaptar Mixtral y BERT se considera la más adecuada para el desarrollo del asistente inteligente, ya que permite un equilibrio óptimo entre la eficiencia, la precisión, la flexibilidad y la capacidad de mejora continua. Se espera que esta estrategia conduzca a la creación de un asistente inteligente, accesible y fácil de usar para cualquier usuario, además, que satisfaga las necesidades del Canal de Isabel II y mejore significativamente la atención al cliente y la resolución de problemas técnicos en el sector del agua.

1.4 Planificación del Trabajo

A. Recursos necesarios

1. Hardware:

- Ordenador personal: HP All-in- One (Procesador AMD Ryzen 5, RAM 16GB).

2. Software:

- Modelos preentrenados de lenguaje natural (BERT, Mixtral).
- Sistema operativo: Windows 11 Home.
- Lenguaje de programación: Python.
- Entornos de desarrollo integrados (IDE): Visual Studio Code.
- Herramientas de control de versiones: GitHub.
- Software de desarrollo:
 - Jupyter: Entorno interactivo de código abierto que soporta Python.
 - Google Drive (Colaboratory).
 - TensorFlow, PyTorch...
 - Librerías de PLN (transformers, datasets, etc.).
 - Librerías para el análisis de datos (pandas, sklearn, etc.).
- Aplicaciones informáticas:
 - Herramientas de Microsoft Office (Word, PowerPoint).
 - Herramientas de gestión de proyectos: *Microsoft Project* (MSP): Administración del proyecto y diagrama de Gantt.
- Plataformas:
 - HuggingFace: Plataforma de desarrollo de herramientas, recursos y modelos de lenguaje para el aprendizaje automático.
 - Gradio.app: Plataforma que ofrece una interfaz gráfica y herramientas para construir y desplegar fácilmente aplicaciones web interactivas.
 - Acceso a plataformas de entrenamiento de modelos (Google Colab).

3. Datos:

- Conjunto de datos de preguntas y respuestas del Canal de Isabel II.
- Documentación técnica del Canal de Isabel II.
- Recursos textuales del sector del agua (documentación técnica, artículos científicos, etc.).

B. Tareas y planificación temporal

PEC 0 (Definición de los contenidos del trabajo)

- Tarea: Elaboración de la Propuesta del TFG.
- Duración: 2 semanas (del 28/02/2024 al 11/03/2024).
- Hito parcial: Entrega de la Propuesta de TFG (11/03/2024).

PEC 1 (Plan de Trabajo)

- Tarea: Elaboración del Plan de Trabajo del TFG.
- Duración: 2 semanas (del 12/03/2024 al 26/03/2024).
- Hito parcial: Entrega del Plan de Trabajo (26/03/2024).
- Hito parcial tareas: Investigar la aplicabilidad de modelos preentrenados (Inicio: 26/03/2024).

PEC 2 (Desarrollo del trabajo: Primer informe de seguimiento)

- Tareas:
 -  Revisión bibliográfica sobre modelos preentrenados.
 -  Selección y análisis de un modelo preentrenado para fine-tuning.
 -  Recopilación y análisis del conjunto de datos.
 -  Redacción del Primer informe de seguimiento.
- Duración: 5 semanas (del 27/03/2024 al 03/05/2024).
- Hito parcial tareas: Realizar fine-tuning e inferencia en modelos preentrenados (Inicio: 26/04/2024).
- Hito parcial tareas: Desarrollo del Asistente (Inicio: 03/05/2024).
- Hito parcial: Entrega del primer informe de seguimiento (03/05/2024).

PEC 3 (Desarrollo del trabajo: Segundo informe de seguimiento)

- Tareas:
 -  Implementación del fine-tuning e inferencia del modelo preentrenado.
 -  Implementación del asistente inteligente.
 -  Diseño de la interfaz del asistente.
 -  Recopilación de datos para la evaluación del asistente.
 -  Redacción del segundo informe de seguimiento.

- Duración: 4 semanas (del 04/05/2024 al 29/05/2024).
- Hito parcial: Entrega del segundo informe de seguimiento (29/05/2024).
- Hito parcial tareas: Desarrollo del Asistente (Inicio: 29/05/2024).

PEC 4 (Redacción de la memoria del trabajo final)

- Tareas:
 - Introducción.
 - Desarrollo.
 - Resultados obtenidos.
 - Conclusiones.
 - Glosario.
 - Bibliografía.
 - Anexos.
- Duración: 2 semanas (del 30/05/2024 al 16/06/2024).
- Hito parcial tareas: Resultados y productos obtenidos (Inicio: 08/06/2024).
- Hito parcial: Entrega de la memoria del TFG (16/06/2024).

PEC 5a (Elaboración de la Presentación del Trabajo)

- Tarea: Elaboración de la presentación del TFG.
- Duración: 1 semana (del 17/06/2024 al 23/06/2024).
- Hito parcial: Entrega de la presentación del TFG (23/06/2024).

PEC 5b (Defensa Pública del Trabajo)

- Tareas:
 - Defensa pública del TFG ante el tribunal.
 - Entrega del informe de autoevaluación.
- Duración: 1 semana (del 24/06/2024 al 30/06/2024).
- Hito parcial: Defensa pública del TFG (30/06/2024).

C. Diagrama de Gantt:

100 /24 | 04 mar '24 | 11 mar '24 | 18 mar '24 | 25 mar '24 | 01 abr '24 | 08 abr '24 | 15 abr '24 | 22 abr '24 | 29 abr '24 | 06 may '24 | 13 may '24 | 20 may '24 | 27 may '24 | 03 jun '24 | 10 jun '24

Agrupar tareas con fechas a la línea de tiempo

Nombre de tarea	Dur.	Comienzo	Fin	Pre	26	02	07	12	17	22	27	01	06	11	16	21	26	01	06	11	16	21	26	31	05	10	15	20	25
▼ Gestión del proyecto TFG.	89 días	mié 28/02/24	dom 30/06/24																										
TFG 0. Elaboración de la Propuesta del TFG.	8 días	jue 29/02/24	dom 10/03/24																										
Entrega PEC0	0 días	lun 11/03/24	lun 11/03/24																										
▼ Desarrollo tareas del TFG.	70 días	mar 12/03/24	dom 16/06/24																										
TFG 1. • Elaborar Plan de Trabajo.	5 días	mié 13/03/24	mar 19/03/24																										
TFG 2. • Desarrollar Diagrama de Gantt • Redactar primera parte de la memoria del trabajo.	4 días	mié 20/03/24	lun 25/03/24																										
Entrega PEC1 - Investigar la aplicabilidad de modelos preentrenados.	0 días	mar 26/03/24	mar 26/03/24																										
TFG 3. • Revisar la literatura existente sobre el uso de modelos preentrenados.	6 días	mié 27/03/24	mié 03/04/24																										
TFG 4. • Análisis de modelos preentrenados para fine-tuning.	5 días	jue 04/04/24	mié 10/04/24																										
TFG 5. • Recopilación y análisis del conjunto de datos.	5 días	jue 11/04/24	mié 17/04/24																										
TFG 6. • Seleccionar modelos preentrenados adecuados para fine-tuning.	3 días	jue 18/04/24	lun 22/04/24																										
TFG 7. • Definir los parámetros y características específicas del fine-tuning para adaptar el modelo al contexto del Canal de Isabel II.	3 días	mar 23/04/24	jue 25/04/24																										
Realizar fine-tuning e inferencia en modelos preentrenados	0 días	vie 26/04/24	vie 26/04/24																										
TFG 8. • Seleccionar un modelo preentrenado adecuado para ajustar parámetros de generación de texto o realizar tareas de PNL para una inferencia específica en el contexto del Canal de Isabel II.	5 días	vie 26/04/24	jue 02/05/24																										
Entrega PEC2 - Desarrollo del Asistente	0 días	vie 03/05/24	vie 03/05/24																										
TFG 9. • Implementar el asistente inteligente utilizando un modelo preentrenado ajustado.	9 días	vie 03/05/24	mié 15/05/24																										
TFG 10. • Diseñar la interfaz del asistente para facilitar la interacción de los clientes o técnicos del Canal de Isabel II.	5 días	jue 16/05/24	mié 22/05/24																										
TFG 11. • Incorporar capacidades de ofrecer soluciones técnicas especializadas a consultas específicas del sector de agua.	4 días	jue 23/05/24	mar 28/05/24																										
Entrega PEC3 - Evaluación del Asistente.	0 días	mié 29/05/24	mié 29/05/24																										
TFG 12. • Recopilación de datos para la evaluación del asistente.	2 días	mié 29/05/24	jue 30/05/24																										
TFG 13. • Valorar la precisión de las respuestas.	2 días	sáb 01/06/24	lun 03/06/24																										
TFG 14. • Evaluar y comparar las respuestas con otros LLM.	2 días	mar 04/06/24	mié 05/06/24																										
TFG 15. • Identificar áreas de mejora.	2 días	jue 06/06/24	vie 07/06/24																										
Resultados y productos obtenidos	0 días	sáb 08/06/24	sáb 08/06/24																										
TFG 16. • Despliegue de los asistentes en la nube con acceso a cualquier usuario.	6 días	sáb 08/06/24	vie 14/06/24																										
Entrega PEC4	0 días	dom 16/06/24	dom 16/06/24																										
▼ UOC	89 días	mié 28/02/24	dom 30/06/24																										

1.5 Breve resumen de productos obtenidos.

- ✚ Notebook de la extracción y preprocesamiento de los datos el texto de una URL.
url = <https://oficinavirtual.canaldeisabelsegunda.es/preguntas-frecuentes>.
- ✚ Conjunto de datos en formato JSON que contiene preguntas y respuestas del Canal de Isabel II.
- ✚ Notebook del proyecto para el Fine-Tuning del modelo BERT pre-entrenado.
- ✚ Archivos que conforman el modelo BERT con Fine-Tuning, incluyendo:
 - config.json
 - model.safetensors
 - special_tokens_map.json
 - tokenizer.json
 - tokenizer_config.json
 - training_args.bin
 - vocab.txt
- ✚ Acceso al interfaz de los asistentes en la plataforma Hugging Face, que incluye:
 - Preguntas y respuestas a usuarios utilizando el modelo BERT con Fine-Tuning.
 - Preguntas y respuestas a usuarios utilizando el modelo Mixtral mediante inferencia.
 - Asistencia de experto en el sector del agua utilizando el modelo Mixtral mediante inferencia.
- ✚ Pruebas de entrenamiento de diferentes versiones.

1.6 Breve descripción de los otros capítulos de la memoria.

Desarrollo:

- Investigación sobre modelos preentrenados: Se realiza una revisión de la literatura sobre modelos preentrenados de lenguaje natural, incluyendo sus características, ventajas y desventajas.
- Recopilación y análisis del conjunto de datos: El dataset que utilizaremos contendrá información relevante de la oficina virtual del Canal de Isabel II, específicamente diseñado para resolver consultas de los usuarios.
- Selección de un modelo preentrenado: Se selecciona un modelo preentrenado adecuado para el proyecto, considerando las necesidades del sector del agua y las características de los modelos disponibles.
- Fine-tuning del modelo preentrenado: Se ajusta el modelo preentrenado al dominio del sector del agua utilizando un conjunto de datos de preguntas y respuestas del Canal de Isabel II.
- Desarrollo del asistente inteligente: Se implementa el asistente inteligente utilizando el modelo preentrenado ajustado, incluyendo la interfaz de usuario y las funcionalidades de atención al cliente.

Relación con el trabajo en global:

- La investigación sobre modelos preentrenados y la selección del modelo adecuado son pasos previos para el fine-tuning y el desarrollo del asistente.
- El dataset estará en formato JSON y contendrá preguntas y respuestas organizadas. El contexto y la pregunta son las entradas al modelo que será entrenado para encontrar la respuesta.
- El fine-tuning permite adaptar el modelo a las necesidades específicas del proyecto, mientras que el desarrollo del asistente integra el modelo en un sistema completo con funcionalidades para la atención al cliente.
- El desarrollo del asistente inteligente es la parte central del proyecto.

Evaluación:

- **Metodología de evaluación:** Se define la metodología que se utilizará para evaluar el rendimiento del asistente inteligente. Se analizará la exactitud de las respuestas generadas por el asistente en comparación con la información real.
- **Resultados de la evaluación:** Se presentan los resultados de la evaluación del asistente inteligente. Se realizarán pruebas con otros modelos para evaluar una comparativa en función de la precisión de las respuestas.
- **Conclusiones y recomendaciones:** Se extraen conclusiones sobre el rendimiento del asistente inteligente y se proponen recomendaciones para su mejora en futuras investigaciones.

Relación con el trabajo en global:

- La evaluación es la fase final del proyecto.
- Los resultados de la evaluación permiten determinar si el asistente inteligente cumple con los objetivos establecidos y si es una solución viable para la atención al cliente en el sector del agua.
- Las conclusiones y recomendaciones de la evaluación son importantes para la mejora del asistente en futuras investigaciones y para la toma de decisiones sobre su implementación en el Canal de Isabel II.

Resultados, productos obtenidos y Conclusiones:

- Se resumen los resultados del proyecto y se analiza la contribución del asistente inteligente a la mejora de la atención al cliente en el sector del agua.
- Se discuten las limitaciones del proyecto y se proponen líneas de investigación futuras para la mejora del asistente inteligente.

Relación con el trabajo en global:

- Las conclusiones del trabajo sintetizan los resultados del proyecto y su impacto en el sector del agua.
- La discusión de las limitaciones y las líneas de investigación futuras son importantes para la planificación de futuros proyectos en esta área.

2. Investigar la aplicabilidad de modelos preentrenados

2.1. Revisar la literatura existente sobre el uso de modelos preentrenados

Problema por resolver

El problema que enfrentamos es el de comprender el contenido de un texto y poder responder preguntas sobre ese texto. Esta tarea, conocida como Preguntas y Respuestas (Q&A) en el procesamiento del lenguaje natural, implica presentar al modelo un contexto (un párrafo de texto) y una pregunta sobre ese contexto, y que el modelo pueda encontrar la porción del texto que responde a esa pregunta.

Modelo para implementar

Utilizaremos BERT como base para nuestro modelo. BERT (*Bidirectional Encoder Representations from Transformers*) es un modelo de lenguaje preentrenado que ha demostrado un rendimiento excepcional en una variedad de tareas de procesamiento del lenguaje natural. Emplearemos un modelo preentrenado de BERT y lo adaptaremos para realizar la tarea de Q&A.

Hugging Face

Hugging Face es una compañía que ha creado un extenso repositorio de modelos de lenguaje basados en transformers, así como variantes preentrenadas de estos modelos. Este repositorio es una invaluable fuente de recursos para la comunidad de procesamiento del lenguaje natural, proporcionando acceso a modelos de última generación y facilitando su implementación y uso en diferentes aplicaciones.

Implementación del modelo

Detallaremos la arquitectura del modelo que vamos a implementar. Utilizaremos un modelo preentrenado de BERT, que ha sido entrenado en un corpus masivo de texto, como base para nuestro modelo. A este modelo preentrenado le realizaremos un *Fine-Tuning* para adaptarlo a la tarea específica de Q&A.

Representación vectorial de las palabras (*word embeddings*) ^[1]

El uso de modelos preentrenados como BERT en aplicaciones de Procesamiento del Lenguaje Natural (PLN), como los sistemas de pregunta-respuesta, ha revolucionado la forma en que representamos el significado de las palabras y realizamos tareas de PLN de alta complejidad.

Las técnicas de *word embeddings*, como *Word2Vec*, *GloVe* y *FastText*, han sido fundamentales para representar el significado de las palabras en forma de vectores. Estas técnicas, basadas en el contexto de las palabras, han evolucionado con modelos *transformers* como BERT, que utilizan el concepto de atención para vectorizar las palabras. BERT mejora significativamente la calidad de los sistemas de traducción automática y los sistemas de pregunta-respuesta al capturar cómo las palabras en una frase contribuyen al significado de una palabra específica. Por ejemplo, en una tarea de traducción automática, BERT representa las palabras en un espacio vectorial donde la similitud entre vectores se calcula utilizando métricas como la *cosine similarity*.

Transfer learning ^{[1] [3]}

El proceso de *transfer learning* permite importar modelos preentrenados, como BERT, que han sido entrenados en grandes cantidades de datos y requieren un costo computacional significativo. Luego, se agrega una capa adicional a estos modelos preentrenados para adaptarlos a tareas específicas, como preguntas-respuestas. Este enfoque reduce el tiempo y los recursos necesarios para entrenar modelos desde cero y permite aprovechar el conocimiento previo aprendido por los modelos preentrenados.

En resumen, implementar un modelo preentrenado de BERT con preguntas y respuestas implica aprovechar las representaciones vectoriales avanzadas de las palabras y utilizar *transfer learning* para adaptar el modelo a la tarea específica de pregunta-respuesta, lo que permite una implementación eficiente y efectiva en aplicaciones de PLN.

¿Cómo funciona BERT? [2] [8] [9]

BERT se sustenta en una robusta arquitectura de red neuronal denominada Transformers. Esta arquitectura integra un mecanismo que capacita a BERT para ponderar el significado de cada término en función de su contexto inmediato, tanto anterior como posterior. Esta habilidad para comprender el contexto dota a BERT de la capacidad para generar incrustaciones de palabras contextualizadas, representaciones que consideran el significado de las palabras en el contexto de las oraciones. Se asemeja a BERT al leer y releer la oración para obtener una comprensión profunda del papel de cada término.

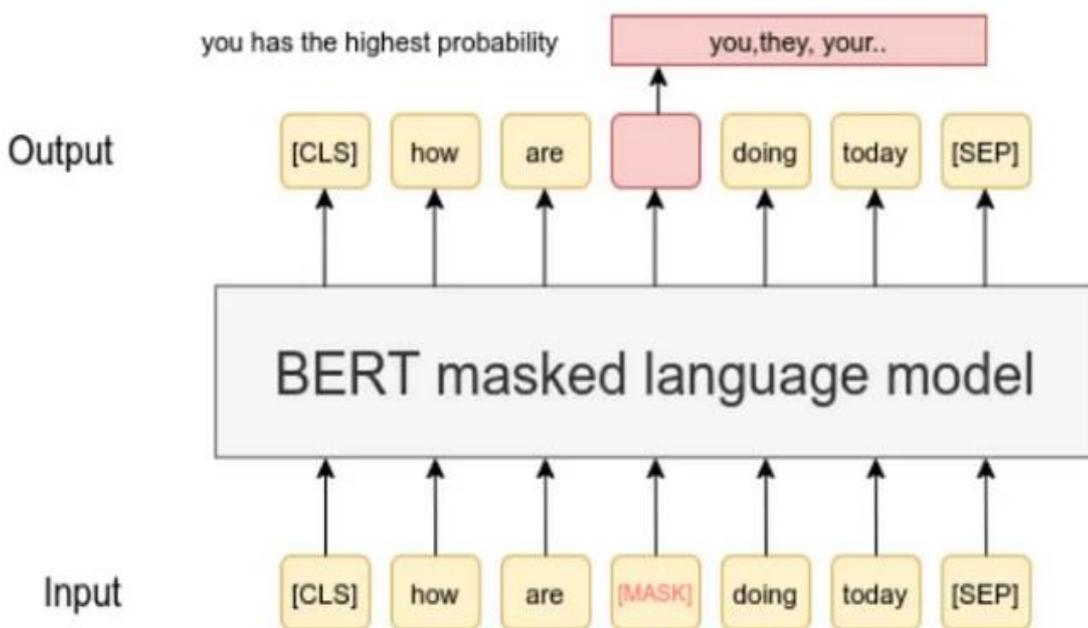


Imagen 1: Analyzing Semantic Equivalence of Sentences Using BERT (analyticsvidhya.com)
<https://www.analyticsvidhya.com/blog/2022/02/analyzing-semantic-equivalence-of-sentences-using-bert/>

Preprocesamiento de datos ^[5] ^[6]

Antes de alimentar los datos al modelo BERT hay que preparar y estructurar el texto de una manera que sea comprensible para él. Los pasos cruciales del preprocesamiento de texto para BERT son la tokenización, el formato de entrada y el propósito del Modelo de Lenguaje Enmascarado (MLM).

- *Tokenización*: Fragmentación del texto en unidades significativas (tanto el contexto como la pregunta).
- *Formato de entrada*: Suministro de contexto a BERT. Para ello, formateamos los tokens de una manera que BERT pueda comprender. Introducimos tokens especiales como [CLS] (indicando clasificación) al principio y [SEP] (indicando separación) entre las oraciones. También asignamos incrustaciones de segmentos para indicar a BERT qué tokens pertenecen a qué oración.
- *Objetivo del Modelo de Lenguaje Enmascarado (MLM)*: Enseñar al contexto de BERT. El secreto de BERT radica en su habilidad para comprender el contexto bidireccional. Durante su entrenamiento, algunas palabras se enmascaran (se reemplazan con [MÁSCARA]) en las oraciones, y BERT aprende a predecir esas palabras a partir de su contexto. Esto ayuda a BERT a captar cómo las palabras se relacionan entre sí, tanto en el contexto anterior como posterior.

En resumen, este enfoque nos permite implementar un modelo de Q&A utilizando BERT, aprovechando el poder de los modelos preentrenados y la facilidad de uso proporcionada por Hugging Face. Mediante el preprocesamiento adecuado de los datos y la adaptación del modelo preentrenado con un *Fine-Tuning*, se va a desarrollar un sistema robusto para responder preguntas sobre un contexto dado.

2.2. Análisis de modelos preentrenados para fine-tuning ^{[10] [11] [12] [24]}

Los modelos de respuesta a preguntas pueden recuperar la respuesta a una pregunta de un texto determinado, lo que resulta útil para buscar una respuesta en un documento. Algunos modelos de respuesta a preguntas pueden generar respuestas sin contexto.

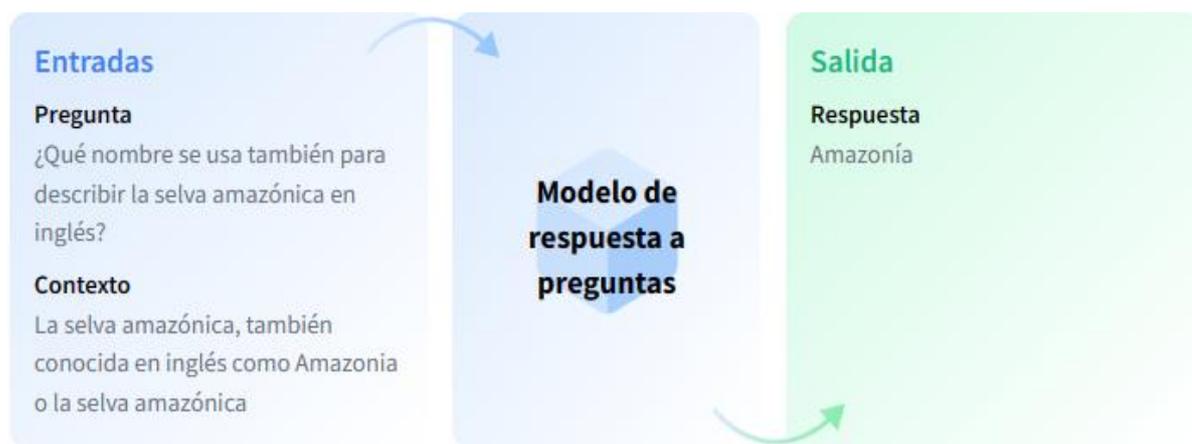


Imagen 2: Esquema de modelo de respuesta a preguntas. <https://huggingface.co/tasks/question-answering>

Modelos de Huggingface/Transformers:

<https://github.com/huggingface/transformers>

Contexto de prueba, común para el análisis de los modelos seleccionados:

```
context = '''
Canal de Isabel II Servicios Canal de Isabel II Servicios es una empresa líder en el sector de servicios de aba
Como entidad pública comprometida con la excelencia y el bienestar de la sociedad, Canal de Isabel II Servicios
a agua potable y la gestión eficiente de los recursos hídricos. Nuestra Misión: La misión de Canal de Isabel II
todos los habitantes de la Comunidad de Madrid. Nos esforzamos por ser líderes en innovación y eficiencia en la
Nuestros Servicios: Abastecimiento de Agua Potable: Garantizamos el suministro constante de agua potable a hoga
Nuestra infraestructura de última generación y nuestras prácticas de gestión avanzada nos permiten cumplir con
Tratamiento de Aguas Residuales: Implementamos sistemas de tratamiento avanzados para asegurar la depuración ef
Contribuimos activamente a la preservación del medio ambiente mediante procesos de tratamiento respetuosos y la
Gestión Integral del Ciclo del Agua: Desde la captación hasta el tratamiento y la distribución, nos encargamos
Gestionamos de manera integral los recursos hídricos, asegurando una planificación eficiente y sostenible.
Innovación y Tecnología: Estamos a la vanguardia de la innovación en el sector del agua.
Implementamos tecnologías emergentes para mejorar la eficiencia operativa, monitorear la calidad del agua en ti
Compromiso Social y Ambiental: Canal de Isabel II Servicios está comprometida con la responsabilidad social y a
Fomentamos prácticas sostenibles, promovemos la educación ambiental y participamos activamente en iniciativas c
Contáctenos: Siempre estamos disponibles para atender las necesidades hídricas de la Comunidad de Madrid.
Canal de Isabel II Servicios se enorgullece de ser un referente en la gestión del agua y está listo para colabo
'''
```

Imagen 3: Contexto de prueba, común para el análisis de los modelos seleccionados

Modelo BERT preentrenado con TensorFlow para preguntas y respuestas

https://huggingface.co/jomacgo/tfm_bert_qa_tf_spanish_model

```
from transformers import BertTokenizer, TFBertForQuestionAnswering

tokenizer = BertTokenizer.from_pretrained('jomacgo/tfm_bert_qa_tf_spanish_model')
model = TFBertForQuestionAnswering.from_pretrained('jomacgo/tfm_bert_qa_tf_spanish_model')
```

```
question = "¿Cuál es el compromiso social de la empresa?"

encoding = tokenizer.encode_plus(text=question, text_pair=context, padding='max_length', max_length=128,

inputs = encoding['input_ids'] #Token embeddings
sentence_embedding = encoding['token_type_ids'] #Segment embeddings
tokens = tokenizer.convert_ids_to_tokens(inputs.numpy()[0]) #input tokens

result = model(input_ids=inputs, token_type_ids=sentence_embedding)

start_logits = result.start_logits.numpy()
end_logits = result.end_logits.numpy()

start_index = np.argmax(start_logits)
end_index = np.argmax(end_logits)

answer = ' '.join(tokens[start_index:end_index+1])
```

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with

```
print(answer.capitalize())
```

Entidad pública comprometida con la excelencia y el bienestar de la sociedad , canal de isabel ii servic

Imagen 4: https://colab.research.google.com/drive/1I9EVUnrh096_R5txRLF7MQcYc3aDGJYd

El modelo `jomacgo/tfm_bert_qa_tf_spanish_model` ha sido entrenado usando TensorFlow.

 `tf_model.h5`. Tamaño: 267 Mb.

Modelo BERT preentrenado con PyTorch para preguntas y respuestas

<https://huggingface.co/dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa>

```
from transformers import BertTokenizer, TFBertForQuestionAnswering

tokenizer = BertTokenizer.from_pretrained('dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa')
model = TFBertForQuestionAnswering.from_pretrained('dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa', from_pt=True)

question = "¿Cuál es el compromiso social de la empresa?"

encoding = tokenizer.encode_plus(text=question, text_pair=context, padding='max_length', max_length=128

inputs = encoding['input_ids'] #Token embeddings
sentence_embedding = encoding['token_type_ids'] #Segment embeddings
tokens = tokenizer.convert_ids_to_tokens(inputs.numpy()[0]) #input tokens

result = model(input_ids=inputs, token_type_ids=sentence_embedding)

start_logits = result.start_logits.numpy()
end_logits = result.end_logits.numpy()

start_index = np.argmax(start_logits)
end_index = np.argmax(end_logits)

answer = ' '.join(tokens[start_index:end_index+1])

Be aware, overflowing tokens are not returned for the setting you have chosen, i.e. sequence pairs with

print(answer.capitalize())
```

Excelencia

Imagen 5: https://colab.research.google.com/drive/1_T5Wjn1LMj74Q1S9wlhwApnP5dViOGwN

Para cargar el modelo hay que tener en cuenta la librería que se usó para entrenarlo. En este caso se usó PyTorch, que se puede ver en la descripción del modelo. Si se usa PyTorch se debe añadir el parámetro `from_pt=True` al cargar el modelo.

 pytorch_model.bin. Tamaño: 437 Mb.

Modelo DistilBERT en idioma inglés para tareas posteriores de preguntas y respuestas

<https://huggingface.co/distilbert/distilbert-base-cased-distilled-squad>

DistilBERT es un modelo de Transformer, pequeño, rápido, barato y ligero; entrenado mediante la base BERT. Tiene un 40% menos de parámetros que *bert-base-uncased*, se ejecuta un 60% más rápido y conserva más del 95% de los rendimientos de BERT.

```
# Creamos el tokenizador con el objeto {"use_fast": False}
nlp = pipeline(
    'question-answering',
    model='distilbert/distilbert-base-cased-distilled-squad',
    tokenizer=(
        'distilbert/distilbert-base-cased-distilled-squad',
        {"use_fast": False}
    )
)
```

```
nlp(
    {
        'question': '¿Cuál es el compromiso social de la empresa?',
        'context': context
    }
)

{'score': 0.04187145456671715,
 'start': 0,
 'end': 57,
 'answer': 'Canal de Isabel II Servicios Canal de Isabel II Servicios'}
```

Imagen 6: <https://colab.research.google.com/drive/1fPRnX6ekoerrvxGoEJk1hWxd1B0vqxU6>

 [model.safetensors](#). Tamaño: 261 Mb.

Modelo BERT preentrenado ajustado (*fine-tuned*) para tareas posteriores de preguntas y respuestas

<https://huggingface.co/mrm8488/longformer-base-4096-spanish-finetuned-squad>

```
from transformers import pipeline
qa_pipe = pipeline("question-answering", model='mrm8488/longformer-base-4096-spanish-finetuned-squad')
```

La función **pipeline** permite crear fácilmente un *pipeline* (tubería) de procesamiento de lenguaje natural para realizar tareas como traducción, resumen de texto, generación de texto, análisis de sentimientos, extracción de entidades, y muchas otras. Simplifica en gran medida el proceso de cargar un modelo preentrenado, tokenizar datos de entrada y ejecutar inferencias.

La función **pipeline** toma como argumento el nombre de la tarea que se quiere realizar, como "*question-answering*" para preguntas y respuestas, "*text-generation*" para generación de texto, "*sentiment-analysis*" para análisis de sentimientos, entre otros. Luego, internamente, carga automáticamente el modelo preentrenado y el tokenizador correspondiente para esa tarea específica.

```
question = "¿Cuál es el compromiso social de la empresa?"

result = qa_pipe({'context':context, 'question': question})
respuesta = result['answer']
print(respuesta.capitalize())
```

Responsabilidad social y ambiental.

Imagen 7: <https://colab.research.google.com/drive/1kUm8VzD2EhYDiYRv9GjXS47ddL6YcktH>

 pytorch_model.bin. Tamaño: 507 Mb.

El contexto tiene 2 posibles respuestas válidas:

- ✓ Como entidad pública comprometida con la excelencia y el bienestar de la sociedad.
- <https://huggingface.co/dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa>
- ✓ Compromiso Social y Ambiental: Canal de Isabel II Servicios está comprometida con la responsabilidad social y ambiental.
- <https://huggingface.co/mrm8488/longformer-base-4096-spanish-finetuned-squad>

2.3. Recopilación y análisis del conjunto de datos ^{[14] [25] [26] [30]}

Los datos etiquetados que utilizaremos contienen información relevante de la oficina virtual del Canal de Isabel II (averías, facturación, bonificaciones, consumos, lecturas, formas de pago, contratación, cambio de titularidad, gestiones, contratos y contacto), específicamente diseñados para resolver consultas de los usuarios. El dataset está en formato JSON y contiene preguntas y respuestas organizadas y de la siguiente manera:

- Cada entrada en el dataset representa una pregunta específica y su correspondiente respuesta.
- Cada entrada contiene un ID único para identificarla de manera única dentro del dataset.
- Además, incluye un título que describa el tema o categoría al que pertenecen las preguntas.
- Se utilizan datos de contexto de la oficina virtual del Canal de Isabel II <https://oficinavirtual.canaldeisabelsegunda.es/preguntas-frecuentes>.
- La pregunta en sí misma está formulada clara y concisamente.
- Las respuestas están estructuradas como un diccionario, donde el texto de la respuesta está acompañado de la posición de inicio de la respuesta dentro del contexto.

Por ejemplo, una entrada en el dataset tiene el siguiente formato:

```
[
  {
    "id" : "100",
    "title" : "Preguntas frecuentes: Averías",
    "context" : "En primer lugar, averigua si.....",
    "question": "¿Qué debo hacer si no tengo agua en mi casa?",
    "answers" : {"text": ["En primer lugar..."], "answer_start": [0]}
  },
  {
    "id" : "101",
    "title" : "Preguntas frecuentes: Averías",
    "context" : "En primer lugar...comprueba que la llave de paso interior...",
    "question": "¿Qué debo revisar si tengo falta de agua en mi vivienda?",
    "answers" : {"text": ["Comprueba que la llave..."], "answer_start": [163]}
  }
]
```

```

1  {
2  }
3  {
4  "id": "100",
5  "title": "Preguntas frecuentes: Averías",
6  "context": "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene
7  "question": "¿Qué debo hacer si no tengo agua en mi casa?",
8  "answers": [{"text": ["En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino
9  ]}],
10 }
11 {
12 "id": "101",
13 "title": "Preguntas frecuentes: Averías",
14 "context": "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene
15 "question": "¿Qué debo revisar si tengo falta de agua en mi vivienda?",
16 "answers": [{"text": ["Comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo
17 }],
18 }
19 {
20 "id": "102",
21 "title": "Preguntas frecuentes: Averías",
22 "context": "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene
23 "question": "¿Dónde puedo encontrar la ubicación del contador?",
24 "answers": [{"text": ["En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador."], "answer_start": [384]}
25 }],
26 }
27 {
28 "id": "103",
29 "title": "Preguntas frecuentes: Averías",
30 "context": "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene
31 "question": "¿Qué debo hacer si sigo sin agua después de verificar el contador?",
32 "answers": [{"text": ["Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros."], "answer_start": [477]}
33 }],
34 }
35 {
36 "id": "104",
37 "title": "Preguntas frecuentes: Averías",
38 "context": "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene
39 "question": "¿Qué debo hacer si aún no tengo agua después de verificar la llave de paso y el contador?",
40 "answers": [{"text": ["Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros."], "answer_start": [477]}

```

Imagen 8: Data Set en formato JSON

3. Realizar fine-tuning e inferencia en modelos preentrenados para adaptarlos al contexto del Canal de Isabel II

3.1. Seleccionar modelos preentrenados adecuados para fine-tuning ^{[15] [21] [27] [28] [29]}

Al elegir entre diferentes modelos BERT para fine-tuning, hay varias opciones que considerar para tomar la mejor decisión:

Tamaño del modelo

- El tamaño del modelo BERT puede variar significativamente y puede afectar el rendimiento y la eficiencia del modelo.
- Los modelos más grandes suelen tener más parámetros y capacidad de representación, lo que puede resultar en un mejor rendimiento en tareas complejas o conjuntos de datos grandes, pero también pueden requerir más recursos computacionales y memoria para entrenar y ejecutar.
- Hay que evaluar las necesidades específicas y los recursos disponibles para determinar si un modelo más grande es adecuado para este caso de uso.

Idioma

- Es crucial elegir un modelo BERT que esté preentrenado en el idioma que se necesita, que es español.

Calidad de las respuestas de prueba

- Antes de elegir un modelo BERT para fine-tuning, es importante evaluar la calidad de las respuestas que proporciona en pruebas de evaluación relevantes.
- Se ha utilizado un contexto de evaluación estándar para comparar el rendimiento de diferentes modelos en tareas específicas de preguntas y respuestas.

Compatibilidad con la tarea específica de preguntas y respuestas

- Hay que considerar si el modelo BERT está preentrenado específicamente para la tarea que se va a realizar o si es más general.
- Algunos modelos BERT pueden estar preentrenados específicamente para tareas de preguntas y respuestas, mientras que otros pueden ser más generales y adaptarse a una variedad de tareas de PLN.
- Se elige un modelo que se adapta mejor a la tarea específica de preguntas y respuestas y que ha demostrado un buen rendimiento en esta área.

Al evaluar estas opciones y consideraciones, podemos seleccionar el modelo BERT más adecuado para el proyecto de fine-tuning y obtener resultados óptimos en las tareas de procesamiento del lenguaje natural.

BETO ^[28]

BETO es un modelo BERT entrenado sobre un gran corpus español. BETO tiene un tamaño similar al de una base BERT y fue entrenado con la técnica de enmascaramiento de palabras completas.

El modelo utiliza un vocabulario de alrededor de 31 mil subpalabras, que fueron construidas utilizando *Byte Pair Encoding* (BPE) con *SentencePiece*. Este vocabulario es utilizado por el modelo para procesar y comprender el texto de entrada. Además, el modelo fue entrenado durante un total de 2 millones de pasos.

- BPE: Codificación de pares de bytes (*Byte Pair Encoding*). Es un algoritmo de compresión de datos y también se utiliza en el procesamiento del lenguaje natural para tokenizar texto.
- *SentencePiece*: Es una biblioteca de tokenización y segmentación de oraciones desarrollada por Google. Se utiliza ampliamente en el procesamiento del lenguaje natural para dividir texto en unidades más pequeñas, como palabras o subpalabras, para su posterior procesamiento por modelos de lenguaje.
- El número de pasos de entrenamiento que ha realizado el modelo durante el proceso de entrenamiento son 2 millones. Cada paso de entrenamiento implica la presentación de una porción de datos al modelo y la actualización de sus parámetros basada en la retroalimentación proporcionada por esos datos. Cuantos más pasos de entrenamiento haya realizado el modelo, en teoría, mejor será su capacidad para comprender y generar texto de manera efectiva.

El modelo "dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa"

Es un modelo BERT preentrenado específicamente para tareas de pregunta y respuesta (Q&A) en español. Algunas características clave sobre este modelo:

- Además de su preentrenamiento en tareas generales de lenguaje, este modelo ha sido afinado (*fine-tuned*) específicamente para tareas de pregunta y respuesta en español.
- Durante el fine-tuning, el modelo ha sido entrenado con conjuntos de datos que contienen ejemplos de preguntas y respuestas en español, lo que le permite comprender mejor el contexto y generar respuestas más precisas.
- El modelo *fine-tuned* ha sido ajustado específicamente para comprender el contexto de las preguntas y generar respuestas relevantes, mientras que el modelo base (*dccuchile/bert-base-spanish-wwm-uncased*) puede no haber sido entrenado específicamente para esta tarea.
- Este modelo ha sido evaluado en el conjunto de datos MLQA (*Multilingual Question Answering*), que es un conjunto de datos de preguntas y respuestas en varios idiomas, incluido el español.
- La evaluación en MLQA proporciona una medida del rendimiento del modelo en tareas de pregunta y respuesta en español, lo que ayuda a validar su precisión y utilidad en aplicaciones del mundo real.

En resumen, el modelo "*dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa*" es una herramienta para el procesamiento del lenguaje natural en español, especialmente en tareas de pregunta y respuesta, gracias a su preentrenamiento en BERT y su afinamiento específico para comprender y generar respuestas precisas en este idioma.

 Repositorio de modelos de HuggingFace:
<https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased#references>

 BETO, Español BERT: <https://github.com/dccuchile/beto>

 Gran Corpus Español: https://huggingface.co/datasets/large_spanish_corpus

3.2. Definir los parámetros y características específicas del fine-tuning para adaptar el modelo al contexto del Canal de Isabel II [4] [11] [13] [18]

Parámetros y características específicas del fine-tuning

Para el modelo "dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa" y considerando hacer fine-tuning para la tarea de preguntas y respuestas con un dataset de aproximadamente 500 preguntas y respuestas etiquetadas, estas son las técnicas aplicadas:

- **Fine-tuning supervisado:** Utilizar el conjunto de datos etiquetado de preguntas y respuestas para ajustar los pesos (los pesos de una red neuronal son los parámetros que se ajustan durante el entrenamiento) del modelo específicamente para esta tarea. Dado que el conjunto de datos es pequeño, esta técnica puede ser efectiva para adaptar el modelo a la tarea específica.
- **Transfer learning:** Aprovechar el conocimiento preentrenado del modelo BERT en el modelado del lenguaje español. Utilizar el modelo preentrenado como punto de partida y ajustar los pesos del modelo para la tarea de preguntas y respuestas utilizando el conjunto de datos etiquetado.
- **Ajuste de hiperparámetros:** Experimentar con diferentes valores de hiperparámetros, como la tasa de aprendizaje, el tamaño del lote, la función de pérdida y el número de épocas de entrenamiento. Realizar una búsqueda exhaustiva de hiperparámetros para encontrar la combinación óptima que maximice el rendimiento del modelo en tu conjunto de datos.

```
# Definir los argumentos de entrenamiento
training_args = TrainingArguments(
    output_dir='./results4',           # Directorio donde se guardarán los resultados del entrenamiento
    num_train_epochs=num_epochs,      # Número de épocas de entrenamiento
    per_device_train_batch_size=batch_size, # Tamaño del lote de entrenamiento por dispositivo
    per_device_eval_batch_size=batch_size, # Tamaño del lote de evaluación por dispositivo
    evaluation_strategy='epoch',      # Estrategia de evaluación
    logging_dir='./logs4',           # Directorio donde se guardarán los logs de entrenamiento
    logging_steps=100,               # Cada cuántos pasos se guardarán los logs
    save_steps=1000,                 # Cada cuántos pasos se guardará el modelo
    save_total_limit=3,              # Límite total de modelos guardados
    learning_rate=lr,                # Tasa de aprendizaje inicial para el optimizador
    weight_decay=weight_decay,       # Decaimiento de peso para evitar sobreajuste
    report_to="all",                 # Reportar todos los resultados, incluida la pérdida de entrenamiento
    save_strategy="epoch",           # Estrategia de guardado del modelo
    load_best_model_at_end=True,     # Cargar el mejor modelo al final del entrenamiento
)
```

Imagen 9: Argumentos que configuran el entrenamiento del modelo BERT

- *evaluation_strategy*: Especifica cuándo se realizará la evaluación durante el entrenamiento. En este caso, “**epoch**” indica que la evaluación se realizará al final de cada época. Esto es adecuado para la mayoría de los casos, especialmente si se tiene un conjunto de datos grande o el entrenamiento es costoso computacionalmente.
- *learning_rate*: La tasa de aprendizaje inicial para el optimizador. Es el tamaño del paso que el modelo toma durante el proceso de optimización para actualizar sus parámetros. La tasa de aprendizaje inicial está establecida en “**1e-5**”. Sin embargo, será beneficioso realizar una búsqueda de tasa de aprendizaje para encontrar el valor óptimo.
- *per_device_train_batch_size*: El tamaño del lote de entrenamiento por dispositivo “**8**”. Indica cuántos ejemplos se procesarán simultáneamente en cada paso de entrenamiento en cada dispositivo. Puede depender de la cantidad de memoria disponible en la GPU y del tamaño del conjunto de datos. Ajustar este valor puede ayudar a optimizar el rendimiento y la velocidad de entrenamiento.
- *per_device_eval_batch_size*: El tamaño del lote de evaluación por dispositivo “**8**”. Similar a *per_device_train_batch_size*, pero para el proceso de evaluación.
- *num_train_epochs*: Entre “**5**” y “**10**”. El número de épocas de entrenamiento, es decir, cuántas veces se iterará sobre todo el conjunto de entrenamiento durante el entrenamiento. Este valor puede variar según la convergencia del modelo y la complejidad del conjunto de datos. A menudo, se utiliza la validación cruzada para determinar el número óptimo de épocas.
- *weight_decay*: Es un término utilizado para prevenir el sobreajuste durante el entrenamiento. Reduce los pesos de los parámetros del modelo durante la optimización, lo que ayuda a evitar que los pesos se vuelvan demasiado grandes. El valor de decaimiento de peso está configurado en “**0.01**”, lo que ayuda a prevenir el sobreajuste durante el entrenamiento. Este valor puede ajustarse según la cantidad de regularización necesaria para el modelo y el conjunto de datos.
- *report_to*: Indica qué resultados deben ser reportados durante el entrenamiento. En este caso, “**all**” significa que se informarán todos los resultados, incluida la pérdida de entrenamiento.

Estos hiperparámetros se utilizan para configurar el proceso de entrenamiento del modelo. Ajustar adecuadamente estos hiperparámetros es crucial para obtener un modelo bien entrenado y con buen rendimiento en la tarea deseada.

- **Data augmentation:** Dado que el conjunto de datos es relativamente pequeño, se considera aplicar técnicas de aumento de datos para aumentar su tamaño y mejorar la capacidad de generalización del modelo. En este caso se aumentará el tamaño del conjunto de datos mediante la reformulación de preguntas y manteniendo el contexto que es una estrategia efectiva.
- **Afinación por capas:** Se agregan capas adicionales al final del clasificador de salida del modelo BERT, con ello estamos extendiendo la arquitectura del modelo al añadir más capas neuronales justo antes de las capas finales que predicen las respuestas, además permite al modelo aprender representaciones más complejas. También se pueden congelar algunas capas del modelo BERT.

Estas técnicas pueden ayudar a realizar un fine-tuning efectivo del modelo junto con el dataset de 500 preguntas y respuestas etiquetadas para la tarea.

Explicación de cada archivo guardado del modelo

1. *config.json*: Este archivo contiene la configuración del modelo, incluyendo los hiperparámetros, la arquitectura y cualquier otra configuración necesaria para recrear el modelo.
2. *model.safetensors*: El archivo contiene los pesos del modelo entrenado. Es esencial para cargar el modelo entrenado en el futuro y realizar inferencias o continuar el entrenamiento desde donde lo dejaste.
3. *special_tokens_map.json*: Este archivo mapea los tokens especiales utilizados por el tokenizador a sus identificadores correspondientes. Estos tokens especiales pueden ser tokens de inicio, tokens de fin, tokens de relleno, etc.
4. *tokenizer.json*: Contiene la información necesaria para recrear el tokenizador utilizado durante el entrenamiento del modelo. Incluye el vocabulario, las configuraciones de tokenización y cualquier otra información relevante.
5. *tokenizer_config.json*: Similar al archivo *tokenizer.json*, este archivo contiene la configuración del tokenizador.
6. *training_args.bin*: El archivo almacena los argumentos de entrenamiento utilizados durante el entrenamiento del modelo. Esto incluye información sobre la configuración del entrenamiento, como la tasa de aprendizaje, el tamaño del lote, el número de épocas, etc.
7. *vocab.txt*: Contiene el vocabulario utilizado por el tokenizador. Esencialmente, mapea los tokens a sus identificadores correspondientes y viceversa. Este archivo se utiliza durante la tokenización del texto de entrada durante la inferencia o el entrenamiento.

El archivo config.json

El archivo *config.json* que se crea al guardar el modelo. Contiene información detallada sobre la arquitectura y la configuración del modelo preentrenado. Explicación detallada de cada uno de los campos:



```
1  {
2    "_name_or_path": "Antonio49/Personal",
3
4    "architectures": [
5      "BertForQuestionAnswering"
6    ],
7    "attention_probs_dropout_prob": 0.1,
8    "classifier_dropout": null,
9    "gradient_checkpointing": false,
10   "hidden_act": "gelu",
11   "hidden_dropout_prob": 0.1,
12   "hidden_size": 768,
13   "initializer_range": 0.02,
14   "intermediate_size": 3072,
15   "layer_norm_eps": 1e-12,
16   "max_position_embeddings": 512,
17   "model_type": "bert",
18   "num_attention_heads": 12,
19   "num_hidden_layers": 12,
20   "output_past": true,
21   "pad_token_id": 1,
22   "position_embedding_type": "absolute",
23   "torch_dtype": "float32",
24   "transformers_version": "4.38.2",
25   "type_vocab_size": 2,
26   "use_cache": true,
27   "vocab_size": 31002
28 }
```

Imagen 10: Archivo config.json

1. *_name_or_path*: Es el nombre o la ruta del modelo preentrenado del que se derivó este modelo específico.
2. *architectures*: Lista de las arquitecturas de los modelos utilizados para preentrenar este modelo específico. En este caso, el modelo se basa en la arquitectura "*BertForQuestionAnswering*".
3. *attention_probs_dropout_prob*: La probabilidad de *dropout* aplicada a los pesos de atención en la capa de atención. *Dropout* es una técnica de regularización que consiste en apagar aleatoriamente un porcentaje de unidades durante el entrenamiento para evitar el sobreajuste.
4. *classifier_dropout*: La probabilidad de *dropout* aplicada a la capa de clasificación. En este caso, es nulo, lo que significa que no se aplica *dropout* a la capa de clasificación.
5. *gradient_checkpointing*: Un indicador de si se utiliza el *checkpointing* de gradientes para reducir el uso de memoria durante el entrenamiento.
6. *hidden_act*: La función de activación utilizada en las capas ocultas del modelo. En este caso, se utiliza la función GELU (*Gaussian Error Linear Unit*).
7. *hidden_dropout_prob*: La probabilidad de *dropout* aplicada a las capas ocultas del modelo.
8. *hidden_size*: El tamaño de la capa oculta del modelo. En este caso, el tamaño es 768.
9. *initializer_range*: El rango de inicialización de los pesos del modelo.
10. *intermediate_size*: El tamaño de las capas intermedias en el modelo.
11. *layer_norm_eps*: La pequeña cantidad *epsilon* agregada a la varianza para estabilizar la normalización por capa.
12. *max_position_embeddings*: La longitud máxima de secuencia permitida para el modelo.
13. *model_type*: El tipo de modelo utilizado. En este caso, es un modelo BERT.
14. *num_attention_heads*: El número de cabezas de atención en la capa de atención.
15. *num_hidden_layers*: El número de capas ocultas en el modelo.
16. *output_past*: Un indicador de si el modelo devuelve el estado pasado. Esto puede ser útil para la generación de texto.
17. *pad_token_id*: El token de relleno utilizado por el modelo.
18. *position_embedding_type*: El tipo de codificación de posición utilizada en el modelo. En este caso, es "*absolute*", lo que significa que se utiliza codificación absoluta de posición.

19. *torch_dtype*: El tipo de datos de los tensores de PyTorch utilizados en el modelo. En este caso, es "float32".
20. *transformers_version*: La versión de la biblioteca Transformers utilizada para entrenar y guardar el modelo.
21. *type_vocab_size*: El tamaño del vocabulario de tipo utilizado por el modelo. En este caso, es 2, lo que indica que el modelo utiliza tokens de tipo para distinguir entre segmentos de texto (por ejemplo, oraciones A y B en el caso de tareas de clasificación de pares de oraciones).
22. *use_cache*: Un indicador de si se utiliza la caché para almacenar los estados intermedios del modelo durante la generación de texto.
23. *vocab_size*: El tamaño del vocabulario utilizado por el modelo. En este caso, es 31002.

El archivo special tokens map.json

El archivo *special_tokens_map.json* contiene información sobre los tokens especiales utilizados en el modelo preentrenado. Explicación detallada de cada campo:

1. *cls_token*: Este campo define el token especial utilizado para el token de clase ([CLS]). El token de clase se utiliza comúnmente en modelos BERT para representar el inicio de una secuencia.
2. *mask_token*: Este campo define el token especial utilizado para el token de máscara ([MASK]). El token de máscara se utiliza en tareas de enmascaramiento de lenguaje como parte del preentrenamiento del modelo.
3. *pad_token*: Este campo define el token especial utilizado para el token de relleno ([PAD]). El token de relleno se utiliza para llenar secuencias a la misma longitud durante el procesamiento por lotes.
4. *sep_token*: Este campo define el token especial utilizado para el token de separación ([SEP]). El token de separación se utiliza en modelos BERT para separar segmentos de texto, por ejemplo, en el contexto de tareas de clasificación de pares de oraciones.
5. *unk_token*: Este campo define el token especial utilizado para el token de desconocido ([UNK]). El token de desconocido se utiliza para representar palabras o caracteres que no están en el vocabulario del modelo.

```
1  {
2    "cls_token": {
3      "content": "[CLS]",
4      "lstrip": false,
5      "normalized": false,
6      "rstrip": false,
7      "single_word": false
8    },
9    "mask_token": {
10     "content": "[MASK]",
11     "lstrip": false,
12     "normalized": false,
13     "rstrip": false,
14     "single_word": false
15   },
16   "pad_token": {
17     "content": "[PAD]",
18     "lstrip": false,
19     "normalized": false,
20     "rstrip": false,
21     "single_word": false
22   },
23   "sep_token": {
24     "content": "[SEP]",
25     "lstrip": false,
26     "normalized": false,
27     "rstrip": false,
28     "single_word": false
29   },
30   "unk_token": {
31     "content": "[UNK]",
32     "lstrip": false,
33     "normalized": false,
34     "rstrip": false,
35     "single_word": false
36   }
37 }
```

Imagen 11: Archivo special_tokens_map.json

El archivo tokenizer_config.json

El archivo *tokenizer_config.json* contiene la configuración del tokenizador utilizado en el modelo. Explicación detallada de cada uno de los campos:

1. *added_tokens_decoder*: Este campo contiene información sobre los tokens especiales añadidos al tokenizador. Cada token especial tiene un índice numérico asociado y se especifica su contenido y propiedades adicionales, como si debe eliminarse de la izquierda (*lstrip*), normalizarse (*normalized*), eliminarse de la derecha (*rstrip*), si consiste en una sola palabra (*single_word*), y si es un token especial (*special*).
2. *clean_up_tokenization_spaces*: Un valor booleano que indica si se deben limpiar los espacios en la tokenización. Cuando es *true*, los espacios redundantes se eliminan después de la tokenización.
3. *cls_token*: El token especial utilizado para el token de clase ([CLS]).
4. *do_basic_tokenize*: Un valor booleano que indica si se debe realizar una tokenización básica. La tokenización básica generalmente implica dividir el texto en palabras y caracteres.
5. *do_lower_case*: Un valor booleano que indica si se debe convertir el texto a minúsculas durante la tokenización. Esto es comúnmente utilizado en modelos que utilizan modelos de lenguaje basados en BERT.
6. *mask_token*: El token especial utilizado para el token de máscara ([MASK]).
7. *model_max_length*: La longitud máxima permitida para las secuencias de entrada al modelo.
8. *never_split*: Una lista de tokens que nunca deben dividirse durante la tokenización.
9. *pad_token*: El token especial utilizado para el token de relleno ([PAD]).
10. *sep_token*: El token especial utilizado para el token de separación ([SEP]).
11. *strip_accents*: Un valor booleano que indica si se deben eliminar los acentos durante la tokenización.
12. *tokenize_chinese_chars*: Un valor booleano que indica si se deben tokenizar los caracteres chinos individualmente. Esto es útil cuando se trabaja con texto chino.
13. *tokenizer_class*: La clase del tokenizador utilizada. En este caso, es "*BertTokenizer*".
14. *unk_token*: El token especial utilizado para el token de desconocido ([UNK]).

En resumen, este archivo contiene la configuración del tokenizador, incluyendo los **tokens especiales**, la configuración de tokenización, y otras opciones relacionadas con el procesamiento de texto.

El archivo tokenizer.json

El archivo *tokenizer.json* contiene la configuración del tokenizador utilizado en el modelo. Explicación detallada de cada uno de los campos:

1. *version*: La versión del archivo de configuración del tokenizador.
2. *truncation*: Configuración para la truncación de secuencias. Esto incluye la dirección de la truncación (*direction*), la longitud máxima de la secuencia (*max_length*), la estrategia de truncación (*strategy*), y el paso de la truncación (*stride*).
3. *padding*: Configuración para el relleno de secuencias. Esto incluye la estrategia de relleno (*strategy*), la dirección de relleno (*direction*), el valor al que se debe rellenar (*pad_id*), el tipo de token de relleno (*pad_type_id*), el token de relleno (*pad_token*), y el múltiplo al que se debe rellenar (*pad_to_multiple_of*).
4. *added_tokens*: Tokens especiales añadidos al tokenizador. Cada token especial tiene un identificador (*id*), contenido (*content*), y propiedades adicionales como si es una sola palabra (*single_word*), si se debe eliminar de la izquierda (*lstrip*), de la derecha (*rstrip*), y si es un token especial (*special*).
5. *normalizer*: Configuración para la normalización del texto. Esto incluye el tipo de normalizador (*type*), si se debe limpiar el texto (*clean_text*), manejar caracteres chinos (*handle_chinese_chars*), eliminar acentos (*strip_accents*), y convertir el texto a minúsculas (*lowercase*).
6. *pre_tokenizer*: Configuración para el pre-tokenizador.
7. *post_processor*: Configuración para el post-procesador. Esto incluye la especificación de tokens especiales como [CLS] y [SEP], y cómo procesar pares de secuencias.
8. *decoder*: Configuración para el decodificador. En este caso, se utiliza el decodificador de *WordPiece*.
9. *model*: Configuración para el modelo de tokenización. Esto incluye el tipo de modelo (*type*), token de desconocido (*unk_token*), prefijo de subpalabra continua (*continuing_subword_prefix*), y la longitud máxima de caracteres de entrada por palabra (*max_input_chars_per_word*), y el vocabulario utilizado.

3.3. Seleccionar un modelo preentrenado adecuado para ajustar parámetros de generación de texto o realizar tareas de procesamiento del lenguaje natural (PLN) para una inferencia específica en el contexto Canal de Isabel II ^[31] ^[32] ^[33]

Los modelos **abiertos** de Mixtral establecen el estándar de eficiencia y están disponibles de forma gratuita, con licencia totalmente permisiva.

A. El modelo Mistral 7B es de despliegue rápido y fácilmente personalizable. Pequeño, pero muy potente para una variedad de casos de uso. Idioma: inglés.

- Mistral-7B-v0.1: Hugging Face
- Mistral-7B-Instruct-v0.2: Hugging Face

B. Los modelos Mixtral 8x7B se han optimizado a través de un ajuste fino supervisado y una optimización directa de preferencias (DPO) para un seguimiento cuidadoso de las instrucciones. Actualmente es uno de los mejores modelos de código abierto, con un rendimiento comparable al de GPT3.5. Idioma: domina el francés, el alemán, el **español**, el italiano y el inglés.

- Mixtral-8x7B-v0.1: Hugging Face
- Mixtral-8x7B-Instruct-v0.1: Hugging Face

Los modelos Mixtral-8x7B-v0.1y y Mixtral-8x7B-Instruct-v0.1 son variantes del *Large Language Model* (LLM) Mixtral-8x7B, un modelo generativo preentrenado basado en una *Sparse Mixture of Experts*. Aquí están las diferencias clave entre ellos:

Mixtral-8x7B-v0.1:

- Este modelo no tiene ninguna restricción de uso y es adecuado para tareas generales de generación de texto.
- No incluye mecanismos de moderación.
- Puede generar respuestas creativas y versátiles.
- No sigue un formato de instrucción específico.

Mixtral-8x7B-Instruct-v0.1:

- Está optimizado para seguir instrucciones específicas.
- Utiliza un formato de instrucción estricto: <s> [INST] Instrucción [/INST] Respuesta del modelo</s>.
- Diseñado para generar salidas óptimas cuando se le proporcionan instrucciones precisas.
- Adecuado para tareas donde se requiere una respuesta específica basada en una instrucción dada.

Para decidir qué modelo es mejor para hacer inferencias, se deben considerar las necesidades específicas y el tipo de datos con los que trabajaremos. Algunos puntos que considerar:

1. Especialización en instrucciones: Si los datos o tareas están específicamente relacionados con instrucciones o guías, es posible que Mixtral-8x7B-Instruct-v0.1 sea más adecuado, ya que este modelo puede estar optimizado para comprender y generar contenido basado en instrucciones.
2. Generalización: Si necesitamos un modelo que pueda manejar una variedad de tareas y datos, es posible que seleccionemos Mixtral-8x7B-v0.1, ya que es un modelo más general sin una especialización específica en instrucciones.
3. Pruebas y experimentación: Se pueden realizar pruebas con ambos modelos para ver cuál produce resultados más satisfactorios para el caso específico de preguntas y respuestas. Esto puede requerir ejecutar varios ejemplos de inferencia y evaluar la calidad de las respuestas generadas.

En resumen, **Mixtral-8x7B-Instruct-v0.1** es más adecuado porque necesitamos que el modelo siga instrucciones específicas para guiar a los clientes o técnicos del Canal de Isabel II.

Parámetros de generación de texto en Mixtral

Mixtral, al igual que otros modelos de lenguaje grande (LLM), ofrece una variedad de parámetros para controlar la generación de texto. Estos parámetros se pueden ajustar para lograr diferentes estilos y resultados. A continuación, se presenta un resumen de los principales parámetros disponibles en Mixtral:

Parámetros básicos:

- **temperature** (temperatura): Controla la "creatividad" del texto generado. Valores más altos (por encima de 1.0) fomentan la generación de texto más diverso e inesperado, mientras que valores más bajos (cerca de 0.0) favorecen la fluidez y la coherencia.
- **top_p** (top p): Controla la diversidad del texto generado. Valores más altos (cerca de 1.0) favorecen la generación de las palabras más probables, mientras que valores más bajos (cerca de 0.0) fomentan la exploración de palabras menos probables.
- **max_new_tokens** (máximo de nuevos tokens): Define la longitud máxima del texto generado.
- **do_sample** (muestreo): Determina si se utiliza el muestreo (True) o el *beam search* (False) para la generación de texto.
- **num_beams** (número de beams): Si se utiliza el beam search, define el número de beams a considerar.
- **num_return_sequences** (número de secuencias de retorno): Define el número de secuencias de texto que se devuelven como resultado.

Parámetros avanzados:

- **bos_token** (token de inicio): Define el token que se utiliza para iniciar la secuencia de texto.
- **eos_token** (token de fin): Define el token que se utiliza para finalizar la secuencia de texto.
- **no_repeat_ngram_size** (tamaño de n-grama no repetitivo): Define la longitud máxima de n-gramas que no se deben repetir en la secuencia de texto.
- **input_max_length** (longitud máxima de entrada): Define la longitud máxima del texto de entrada.
- **output_max_length** (longitud máxima de salida): Define la longitud máxima del texto generado.

- ***encoder_attention_probs_dropout_prob*** (probabilidad de *dropout* en las probabilidades de atención del encoder): Define la probabilidad de *dropout* en las probabilidades de atención del encoder.
- ***decoder_attention_probs_dropout_prob*** (probabilidad de *dropout* en las probabilidades de atención del decoder): Define la probabilidad de *dropout* en las probabilidades de atención del decoder.
- ***generator_probs_dropout_prob*** (probabilidad de *dropout* en las probabilidades de generación): Define la probabilidad de *dropout* en las probabilidades de generación.

Consideraciones

- El ***beam search*** es un algoritmo utilizado en la generación de texto para predecir la siguiente palabra o token en una secuencia de manera más eficiente. En lugar de evaluar todas las posibles secuencias de tokens, lo cual sería computacionalmente costoso y poco práctico, el *beam search* mantiene un conjunto (o haz) de las secuencias más prometedoras en cada paso de la generación.
- Los ***n-gramas*** son secuencias contiguas de *n* elementos, generalmente palabras, dentro de un texto o corpus de texto. La "n" en n-grama representa el número de elementos en la secuencia. Por ejemplo, un 1-grama es simplemente una palabra individual, un 2-grama (también llamado bigrama) es una secuencia de dos palabras consecutivas, un 3-grama (trigrama) es una secuencia de tres palabras consecutivas, y así sucesivamente.
- El ***dropout*** es una técnica de regularización utilizada en redes neuronales para prevenir el sobreajuste (*overfitting*). Consiste en apagar aleatoriamente un porcentaje de las unidades neuronales durante el entrenamiento.
- La elección de los parámetros adecuados depende de la tarea específica y del estilo de texto deseado.
- Es recomendable explorar diferentes parámetros y configuraciones para encontrar la que mejor se adapte a las necesidades de nuestra inferencia.
- Mixtral, al igual que otros LLMs, puede requerir un entrenamiento fino para obtener un rendimiento óptimo en tareas específicas.

4. Desarrollar un asistente que guíe a los clientes y/u ofrezca soluciones técnicas especializadas

4.1. Implementar el asistente inteligente con un modelo preentrenado ajustado ^[6] ^[30]

Preguntas y respuestas utilizando el modelo BERT mediante inferencia.

- <https://antonio49-bertfinetuningcontextopredefinido2.hf.space>
- <https://github.com/antionideepblue/CanalBERTfine-tuning>
- <https://huggingface.co/spaces/Antonio49/BERTfinetuningContextoPredefinido2>



Imagen 12: Modelo BERT.

<https://huggingface.co/spaces/Antonio49/BERTfinetuningContextoPredefinido2>

Despliegue del modelo BERT

El despliegue del modelo BERT preentrenado con fine-tuning específico para preguntas y respuestas utilizando el framework Hugging Face, se concentra en el archivo *app.py*. Este archivo carga el modelo BERT *finetuned* para preguntas y respuestas y proporciona una interfaz de usuario simple para interactuar con el modelo y obtener respuestas en función de las preguntas ingresadas. Se detallan los componentes y funciones clave del archivo:

1. Importación de bibliotecas y módulos:
 - **torch**: Biblioteca de aprendizaje automático de PyTorch.
 - **AutoModelForQuestionAnswering**, **AutoTokenizer**, **pipeline**: Clases y funciones de la biblioteca Transformers de Hugging Face para cargar el modelo y configurar el pipeline de preguntas y respuestas.
 - **gradio**: Biblioteca Gradio para crear interfaces de usuario interactivas.
2. Carga del modelo y el tokenizador:
 - Se especifica el nombre del modelo preentrenado ('Antonio49/ModeloCanal') y se carga tanto el modelo como el tokenizador utilizando la función **from_pretrained** de la clase correspondiente.
3. Configuración del pipeline de pregunta-respuesta:
 - Se utiliza la función **pipeline** de la biblioteca Transformers para configurar el pipeline de pregunta-respuesta con el modelo y el tokenizador cargados anteriormente.
4. Definición del contexto:
 - Se define un contexto predefinido en el código que proporciona información adicional para ayudar al modelo a responder preguntas. Este contexto se utiliza en conjunto con las preguntas para generar respuestas.
5. Función para responder preguntas:
 - Se define una función llamada **answer_question** que toma una pregunta como entrada y devuelve la respuesta generada por el modelo utilizando el contexto definido anteriormente.
6. Interfaz Gradio:
 - Se define la interfaz de usuario utilizando Gradio, donde se proporciona un cuadro de texto para que el usuario ingrese la pregunta y se mostrará la respuesta generada por el modelo.
 - Se especifican metadatos adicionales para la interfaz, como el título, la descripción y el autor del modelo.
7. Lanzamiento de la interfaz:
 - Se utiliza la función **launch()** para iniciar la interfaz Gradio y permitir que los usuarios interactúen con el modelo proporcionando preguntas y obteniendo respuestas en tiempo real.

El código del archivo *app.py* en el framework Hugging Face:

```
# Importar las bibliotecas necesarias
import torch
from transformers import AutoModelForQuestionAnswering, AutoTokenizer,
pipeline
import gradio as gr

# Cargar el modelo y el tokenizador
model_name = 'Antonio49/ModeloCanal'
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Configurar el pipeline para la pregunta-respuesta
nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)

# Contexto integrado en el código
context = "Averías: En primer lugar, averigua si la falta de agua afecta
a todos los puntos de agua de tu casa..."

# Función para responder preguntas dadas una pregunta y un contexto
predefinido
def answer_question(question):

    # Convertir la pregunta a minúsculas
    question = question.lower()

    # Obtener la respuesta del modelo
    result = nlp(question=question, context=context)

    # Verificar si la respuesta supera un cierto score de confianza
    if result["score"] > 0.79:
        # Verificar si se encontró una respuesta
        if result['answer']: # Si la respuesta no está vacía
            # Retornar la respuesta encontrada
            return result['answer']
        else:
```

```
        # Devolver un mensaje indicando que la pregunta debe ser
reformulada
        return "Lo siento, no pude encontrar una respuesta para tu
pregunta. Por favor, reformula tu pregunta sobre Averías o Contacto del
Canal de Isabel II."
    else:
        # Devolver un mensaje indicando que la pregunta debe ser
reformulada
        return "Lo siento, no estoy seguro de la respuesta. Por favor,
reformula tu pregunta sobre Averías o Contacto del Canal de Isabel II."

# Definir la interfaz Gradio
iface = gr.Interface(fn=answer_question,
                    inputs= gr.Textbox(label="Question",
placeholder="Escribe tu pregunta aquí...", scale=7),
                    outputs=gr.Textbox(label="Answer"),
                    theme="soft",
                    title="BERT_fine-tuning: Preguntas y Respuestas del
Canal de Isabel II",
                    description='Autor: <a
href=\\"https://huggingface.co/Antonio49\\">Antonio Fernández</a> de <a
href=\\"https://www.canaldeisabelsegunda.es/\\">Canal de Isabel II</a>.
Formación: <a href=\\"https://www.uoc.edu/es/\\">Grado Ingeniería
Informática</a> Aplicación desarrollada para
TFG_Inteligencia_Artificial.'" Proporcione una pregunta y el asistente
encontrará la respuesta.')

if __name__ == "__main__":
    # Lanzar la interfaz
    iface.launch()
```

Guardar el modelo en Hugging Face

Se crea una tarjeta de modelo que pretende ser una plantilla base para nuevos modelos. Y es el modelo que se utilizará para la aplicación del Asistente.

Existe la posibilidad de utilizar el modelo directamente sobre la tarjeta creada:



The screenshot shows a Hugging Face model card for 'Safetensors'. At the top, it displays 'Safetensors' with a help icon, 'Model size 109M params', 'Tensor type F32', and a share icon. Below this is the 'Inference API' section, which includes a 'Question Answering' task and an 'Examples' dropdown menu. A text input field contains the question '¿Que conceptos tiene la factura?' and a 'Compute' button. Underneath, a 'Context' box contains a paragraph of text about invoice concepts. Below the context, it shows 'Computation time on cpu: 0.071 s' and a result box with the word 'cuatro' and a score of '0.209'. At the bottom, there is a 'JSON Output' section and a 'Maximize' button. The footer of the interface indicates 'Space using Antonio49/Personal 1' and a specific model identifier 'Antonio49/Question-and-Answer-Assistant3'.

Imagen 13: Tarjeta modelo. <https://huggingface.co/Antonio49/Personal>

Safetensors (Tensores de seguridad) Model size 109M params. Tensor type F32.

Safetensors es un nuevo formato simple para almacenar tensores de forma segura (a diferencia de pickle). Safetensors es realmente rápido. Los archivos del modelo se guardan en la tarjeta creada, además los archivos:

- *.gitattributes*: Este archivo es utilizado por el sistema de control de versiones Git para definir cómo deben tratarse ciertos archivos y líneas de texto en el repositorio.
- *README.md*: Es un archivo en formato Markdown (generalmente en un formato que puede ser convertido a HTML) que proporciona información importante sobre el proyecto.

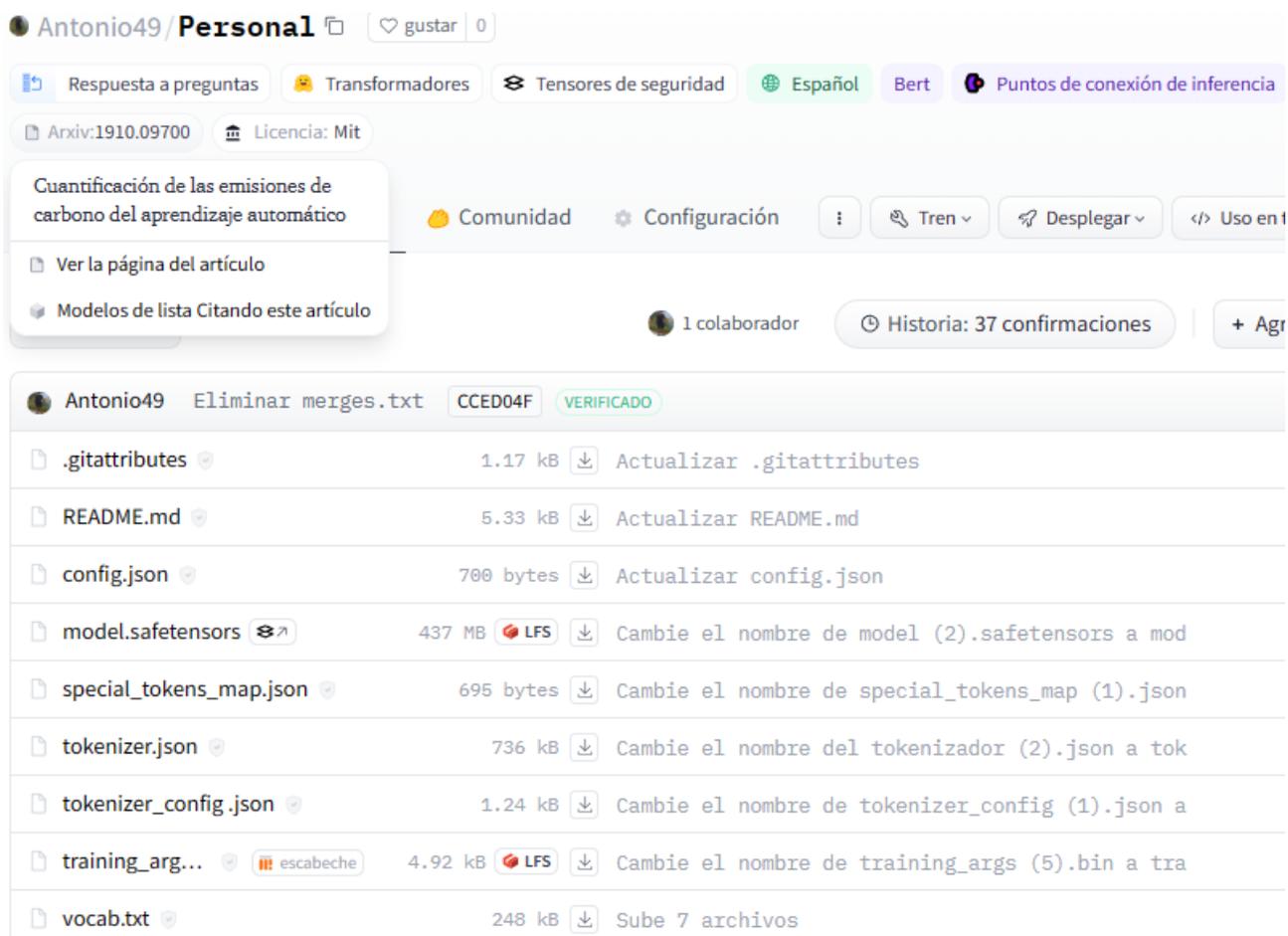


Imagen 14: Archivos de la tarjeta modelo. <https://huggingface.co/Antonio49/Personal/tree/main>

Cómo usar desde el **Transformadores** biblioteca

```
# Use a pipeline as a high-level helper  
from transformers import pipeline
```

```
pipe = pipeline("question-answering", model="Antonio49/Personal")
```

```
# Load model directly
```

```
from transformers import AutoTokenizer, AutoModelForQuestionAnswering
```

```
tokenizer = AutoTokenizer.from_pretrained("Antonio49/Personal")
```

```
model = AutoModelForQuestionAnswering.from_pretrained("Antonio49/Personal")
```

Enlaces rápidos

- 🔗 Leer la documentación del modelo
- 🔗 Leer documentos en canalización de alto nivel
- 🔗 Lea nuestros recursos de aprendizaje

Imagen 15: Acceso al modelo desde transformers. <https://huggingface.co/Antonio49/Personal>

Modelo final: <https://huggingface.co/Antonio49/ModeloCanal>

4.2. Diseñar la interfaz del asistente para facilitar la interacción con los clientes o técnicos del Canal de Isabel II [35] [36] [38]

Preguntas y respuestas utilizando el modelo Mixtral mediante inferencia.

- <https://antonio49-chat-mixtral-aguausuarioconhilos.hf.space>
- <https://github.com/antoniodeepblue/Chat.Mixtral.AguaUsuarioConHilos>
- <https://huggingface.co/spaces/Antonio49/Chat.Mixtral.AguaUsuarioConHilos>



Imagen 16: Modelo Mixtral.

<https://huggingface.co/spaces/Antonio49/Chat.Mixtral.AguaUsuarioConHilos>

Despliegue del modelo Mixtral

El despliegue del modelo Mixtral preentrenado, se concentra en el archivo *app.py*. Este archivo carga un modelo Mixtral preentrenado para realizar inferencias, define funciones para formatear el prompt y generar respuestas, y crea una interfaz de chat interactiva utilizando Gradio para que los usuarios de la empresa Canal de Isabel II puedan interactuar con el modelo y obtener respuestas en tiempo real. La explicación detallada del archivo:

1. Importación de bibliotecas:

- Se importan las bibliotecas necesarias, incluyendo **InferenceClient** de **huggingface_hub** para cargar el modelo preentrenado y **gradio** para crear la interfaz de chat.

2. Creación del cliente de inferencia:

- Se crea un cliente de inferencia utilizando **InferenceClient** de Hugging Face Hub, que se utiliza para cargar el modelo preentrenado **Mixtral-8x7B-Instruct-v0.1**.

3. Función para formatear el prompt:

- Se define una función llamada **format_prompt** para formatear el prompt de entrada, que incluye el historial de interacción con el bot y el prompt del sistema (en este caso, específico para usuarios de Canal de Isabel II).

4. Función para generar respuestas:

- Se define una función llamada **generate** que toma como entrada el prompt, el historial de interacciones, y varios parámetros de configuración para la generación de texto.
- Los parámetros de configuración incluyen *temperature*, *max_new_tokens*, *top_p*, *repetition_penalty*, *input_max_length*, entre otros.
- Se ajustan los valores de *temperature* y *top_p* para asegurar que estén en el rango adecuado.
- Se configuran los parámetros para la generación de texto y se formatea el prompt.
- Se obtiene la respuesta del modelo de manera continua, utilizando el cliente de inferencia y se devuelve la respuesta generada.

5. Creación de la interfaz de chat Gradio:

- Se utiliza **gr.ChatInterface** para crear una interfaz de chat con el modelo generativo.
- Se especifican diferentes componentes de la interfaz, como el cuadro de texto para ingresar preguntas, el tema, el título y la descripción.
- Se define un chatbot con características visuales específicas, como avatares, tamaño de burbujas y diseño.
- Se lanza la interfaz de chat utilizando **launch()**.

```
# Importar las bibliotecas necesarias
from huggingface_hub import InferenceClient
import gradio as gr
# Crear un cliente de inferencia para el modelo preentrenado Mixtral-
8x7B-Instruct-v0.1
client = InferenceClient("mistralai/Mixtral-8x7B-Instruct-v0.1")
# Función para formatear el prompt con historial
def format_prompt(message, history, system_prompt):
    prompt = "<s>"
    for user_prompt, bot_response in history:
        prompt += f"[INST] {user_prompt} [/INST]"
        prompt += f" {bot_response}</s> "
    prompt += f"[INST] {system_prompt}, {message} [/INST]"
    return prompt

# Función para generar respuestas dada una serie de parámetros
def generate(
    prompt,
    history,
    system_prompt= "Asistente para los usuarios y clientes de la empresa
Canal de Isabel II, https://oficinavirtual.canaldeisabelsegunda.es/ ",
    temperature=0.9,
    max_new_tokens=4096,
    top_p=0.95,
    repetition_penalty=1.0,
    input_max_length=256,):
    # Ajustar valores de temperatura y top_p para asegurar que estén en
el rango adecuado
    temperature = float(temperature)
    if temperature < 1e-2:
        temperature = 1e-2
    top_p = float(top_p)

    # Configurar los parámetros para la generación de texto
    generate_kwargs = dict(
        temperature=temperature,
        max_new_tokens=max_new_tokens,
        top_p=top_p,
        repetition_penalty=repetition_penalty,
        do_sample=True,
        seed=42,
    )
```

```
# Formatear el prompt y obtener la respuesta del modelo de manera
continua
formatted_prompt = format_prompt(prompt, history, system_prompt)
stream = client.text_generation(formatted_prompt, **generate_kwargs,
stream=True, details=True, return_full_text=False)
output = ""
# Iterar a través de las respuestas en el stream
for response in stream:
    output += response.token.text
    yield output
return output

# Crear una interfaz de chat Gradio con el modelo generativo
gr.ChatInterface(
    fn=generate,
    chatbot=gr.Chatbot(
        avatar_images=["./15f4b2d3-c4f4-4a29-93cd-e47214953bd9.png",
"./logoCanal.png"],
        bubble_full_width=False,
        show_label=False,
        show_share_button=False,
        show_copy_button=True,
        likeable=True,
        layout="panel",
        height=500,
    ),
    textbox=gr.Textbox(placeholder="¿Qué parámetros definen la calidad
del agua?", container=False, scale=7),
    theme="soft",
    title="Mixtral 8B. TFG: Asistente de usuarios/clientes de Canal de
Isabel II",
    description='Autor: <a
href="https://huggingface.co/Antonio49">Antonio Fernández</a> de <a
href="https://www.canaldeisabelsegunda.es/">Canal de Isabel II</a>.
Formación: <a href="https://www.uoc.edu/es/">Grado Ingeniería
Informática</a>. Aplicación desarrollada para TFG',
        retry_btn="Repetir",
        undo_btn="Deshacer",
        clear_btn="Borrar",
        submit_btn="Enviar",
        concurrency_limit=20,
    ).launch(show_api=False)
```

Función de generación de respuestas del modelo Mixtral seleccionado:

```
# Función para generar respuestas dada una serie de parámetros
def generate(
    prompt,
    history,
    system_prompt= "Asistente para los usuarios y clientes de la empresa Canal de Isabel II
    temperature=0.9,
    max_new_tokens=4096,
    top_p=0.95,
    repetition_penalty=1.0,
    input_max_length=256,):
# Ajustar valores de temperatura y top_p para asegurar que estén en el rango adecuado
    temperature = float(temperature)
    if temperature < 1e-2:
        temperature = 1e-2
    top_p = float(top_p)

# Configurar los parámetros para la generación de texto
    generate_kwargs = dict(
        temperature=temperature,
        max_new_tokens=max_new_tokens,
        top_p=top_p,
        repetition_penalty=repetition_penalty,
        do_sample=True,
        seed=42,
    )
```

Imagen 17: Función de generación de respuestas del modelo Mixtral seleccionado

Cada parámetro en la función de generación de respuestas para el modelo Mixtral:

1. prompt: Es el texto de entrada que proporciona la entrada inicial para la generación de texto. Por lo general, incluirá la pregunta o el contexto sobre el cual se desea generar una respuesta.
2. history: Es el historial de conversación o contexto previo que se proporciona al modelo. Puede incluir mensajes anteriores en una conversación o información contextual relevante para la generación de la respuesta.

3. system_prompt: Es el prompt del sistema, que en este caso es una descripción estática del sistema o del entorno en el que se utiliza el modelo. Este prompt se utiliza para proporcionar información adicional al modelo y consecuentemente ayudarlo en la generación de respuestas.
 - "Asistente para los usuarios y clientes de la empresa Canal de Isabel II, <https://oficinavirtual.canaldeisabelsegunda.es/>".
 - "Experto en servicios de abastecimiento, depuración, reutilización, alcantarillado y calidad del agua, para la empresa Canal de Isabel II. Asistente que ofrece información: obras hidráulicas, cumplimiento de sanidad y seguridad de las instalaciones del Canal de Isabel II".
4. temperature=0.9: Es un parámetro de "temperatura" que controla la aleatoriedad de las predicciones del modelo durante la generación de texto. Una temperatura más alta aumenta la diversidad de las muestras generadas al suavizar las distribuciones de probabilidad de los tokens de salida, lo que significa que se explorarán opciones más diversas. Una temperatura baja, por otro lado, tiende a producir resultados más deterministas y apegados a las predicciones más probables del modelo.
5. max_new_tokens=4096: Este parámetro establece el máximo número de tokens que se generarán como salida durante la generación de texto. Limitar el número de tokens puede ayudar a controlar la longitud de las respuestas generadas y evitar que el modelo genere texto excesivamente largo. Para el proyecto se requiere que las respuestas sean explicativas con detalle y sin restricciones en el límite de palabras (tokens).
6. top_p=0.95: Este parámetro controla la cantidad acumulativa de probabilidad de los tokens considerados durante el muestreo. Específicamente, *top_p* establece la suma acumulativa de las probabilidades de los tokens ordenados en función de su probabilidad descendente. Durante el muestreo, se eligen tokens hasta que la suma acumulativa de probabilidades excede *top_p*. Esto permite controlar la diversidad de las muestras generadas al garantizar que solo se consideren los tokens más probables hasta cierto punto.
7. repetition_penalty=1.0: Es un parámetro que controla la penalización por repetición durante la generación de texto. Cuando *repetition_penalty* es mayor que 1.0, el modelo penaliza más fuertemente la repetición de tokens en la salida generada. Esto ayuda a evitar que el modelo genere respuestas con contenido repetitivo o redundante. Por otro lado, si *repetition_penalty* es menor que 1.0, se fomenta la repetición en la salida generada.

8. input_max_length=256: Con este parámetro se establece la longitud máxima permitida para la entrada de texto que se proporciona al modelo. Limitar la longitud máxima de la entrada puede ser útil para controlar el tiempo de inferencia y evitar problemas de memoria al procesar secuencias de entrada muy largas. Además, puede ayudar a garantizar que el modelo se concentre en la información más relevante de la entrada y evite el procesamiento de texto innecesariamente largo.
9. temperature = float(temperature): Esta línea convierte el valor de *temperature* en un número de punto flotante (float). Aunque se espera que *temperature* sea un float, esta línea garantiza que se convierta explícitamente en caso de que el usuario haya proporcionado un valor diferente.
10. if temperature < 1e-2: temperature = 1e-2: Esta línea verifica si el valor de *temperature* es menor que 0.01 (1e-2 en notación científica). Si es menor que este umbral, se establece *temperature* en 0.01. Esto se hace para evitar valores de *temperature* extremadamente bajos que podrían afectar negativamente la diversidad de las respuestas generadas por el modelo. Una *temperature* muy baja haría que el modelo genere respuestas muy determinísticas y poco variadas.
11. top_p = float(top_p): Esta línea convierte el valor de *top_p* en un número de punto flotante (float), asegurando que sea del tipo de datos adecuado para su uso en la generación de texto.
12. do_sample=True: Este parámetro indica si se debe utilizar el muestreo durante la generación de texto. Cuando *do_sample* está establecido en *True*, el modelo utiliza muestreo durante la generación de texto, lo que significa que elige tokens de salida basados en su probabilidad de ocurrencia en lugar de simplemente seleccionar el token más probable en cada paso. Esto puede conducir a respuestas más diversas y creativas, ya que el modelo tiene la libertad de explorar diferentes opciones durante la generación de texto.
13. seed=42: Este parámetro establece la semilla aleatoria para el generador de números pseudoaleatorios. Establecer una semilla aleatoria garantiza que la generación de texto sea reproducible, es decir, que producirá los mismos resultados cada vez que se ejecute con la misma semilla. Esto es útil para propósitos de depuración y reproducibilidad, ya que permite obtener los mismos resultados cada vez que se ejecuta el código. La elección del número 42 como semilla es arbitraria y se utiliza comúnmente en la comunidad de programación.

4.3. Incorporar capacidades de ofrecer soluciones técnicas especializadas a consultas específicas del sector del agua ^[35] ^[36] ^[38]

Asistencia de experto en el sector del agua utilizando el modelo Mixtral mediante inferencia.

- <https://antonio49-chat-mixtral-expertoconhilos.hf.space>
- <https://github.com/antoniodeepblue/Chat.Mixtral.ExpertoConHilos>
- <https://huggingface.co/spaces/Antonio49/Chat.Mixtral.ExpertoConHilos>

El código sería el mismo que en el punto anterior, salvo que cambiamos el `system_prompt`:

```
# Importar las bibliotecas necesarias
from huggingface_hub import InferenceClient
import gradio as gr

# Crear un cliente de inferencia para el modelo preentrenado Mixtral-8x7B-Instruct-v0.1
client = InferenceClient("mistralai/Mixtral-8x7B-Instruct-v0.1")

# Función para formatear el prompt con historial
def format_prompt(message, history, system_prompt):
    prompt = "<s>"
    for user_prompt, bot_response in history:
        prompt += f"[INST] {user_prompt} [/INST]"
        prompt += f" {bot_response}</s> "
    prompt += f"[INST] {system_prompt}, {message} [/INST]"
    return prompt

# Función para generar respuestas dada una serie de parámetros
def generate(
    prompt,
    history,
    system_prompt="Experto en servicios de abastecimiento, depuración, reutilización, alcantarillado y calidad del agua, para la empresa Canal de Isabel II. Asistente que ofrece información: obras hidráulicas, cumplimiento de sanidad y seguridad de las instalaciones del Canal de Isabel II",
    temperature=0.9,
```



Imagen 18: Modelo Mixtral.
<https://huggingface.co/spaces/Antonio49/Chat.Mixtral.ExpertoConHilos>

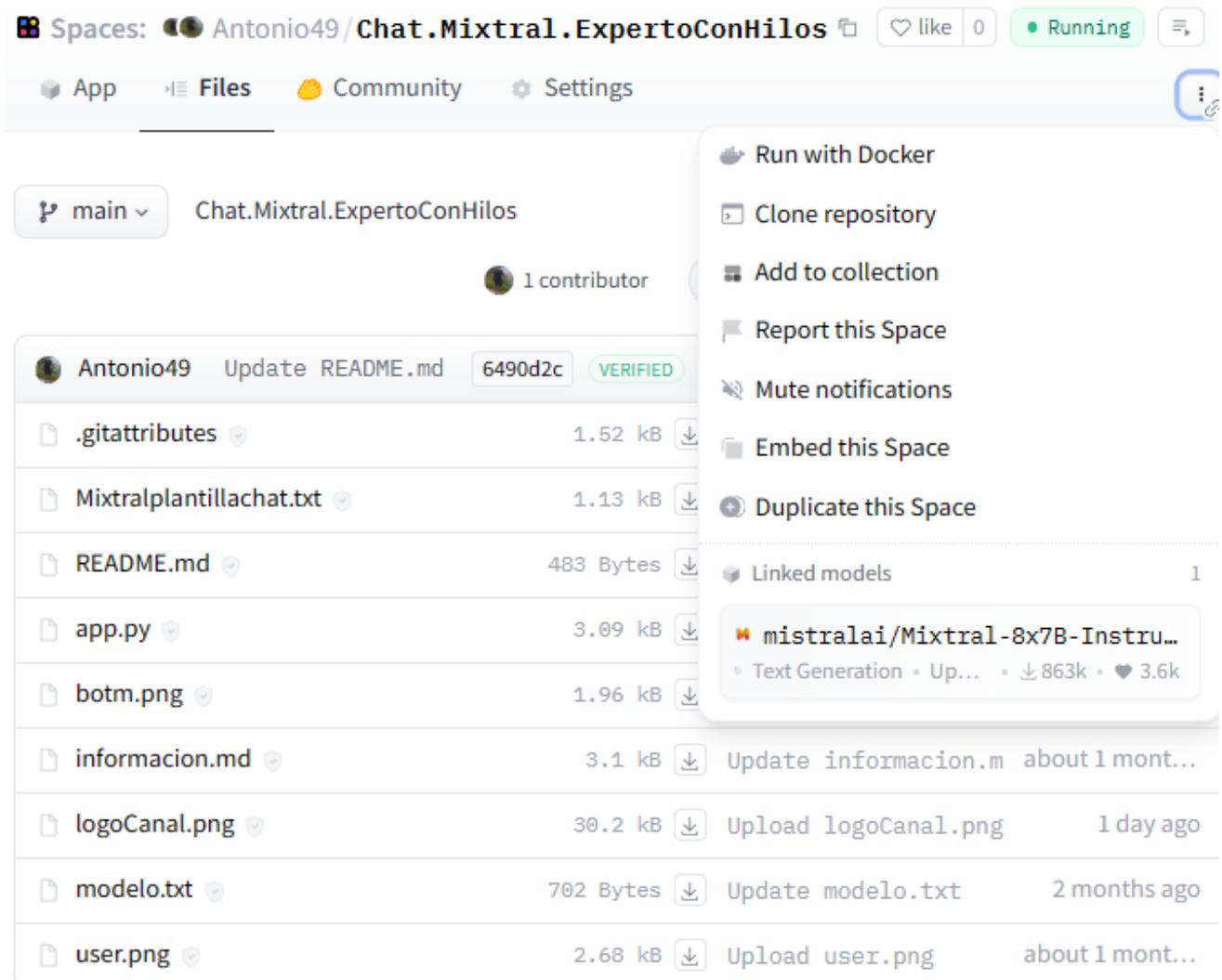


Imagen 19: Archivos del espacio del modelo Mixtral.
<https://huggingface.co/spaces/Antonio49/Chat.Mixtral.ExpertoConHilos/tree/main>

5. Evaluación del asistente

5.1. Recopilación de datos para la evaluación del asistente

Introducción

Se analizarán los aspectos relacionados con el entrenamiento, las métricas utilizadas, el filtrado de respuestas y el rendimiento general del asistente. Se implementarán métricas de evaluación para medir el rendimiento del asistente inteligente en diferentes escenarios.

Entrenamiento

Se realizaron tres entrenamientos del modelo con los mismos parámetros, utilizando diferentes optimizadores: **SGD**, **Adam** y **AdamW**. Se espera que un buen entrenamiento se traduzca en respuestas de calidad y, por lo tanto, en una evaluación favorable.

Métricas léxicas

Para evaluar el rendimiento del asistente, se compararán las respuestas generadas con las respuestas esperadas. Se emplearán diversas métricas, incluyendo:

- *Accuracy*: Exactitud de la respuesta generada en comparación con la respuesta de referencia.
- *Precision*: Porcentaje de palabras correctas en la respuesta generada respecto a la referencia.
- *Recall*: Porcentaje de palabras relevantes de la respuesta de referencia presentes en la respuesta generada.
- *F1-Score*: Media armónica entre *precision* y *recall*.

Selección de métricas

Debido al tamaño del conjunto de datos (500 preguntas-respuestas), se espera que las respuestas sean relativamente precisas. Se analizará la exactitud de las respuestas generadas por el asistente en comparación con la información real. Por lo tanto, se utilizará el *accuracy* como referencia principal, pero también se evaluarán *precision*, *recall* y *F1-Score*.

Cálculo de métricas

- *Precision*: (Palabras coincidentes / Total palabras respuesta generada)
- *Recall*: (Palabras coincidentes / Total palabras respuesta referencia)
- *F1-Score*: $2 * (Precision * Recall) / (Precision + Recall)$
- *Accuracy*: (Respuestas correctas / Total respuestas)

Consideraciones

- No se distingue entre mayúsculas y minúsculas.
- La *precision*, *recall* y *F1-Score* pueden no coincidir con el *accuracy* debido a que no consideran la relación token a token entre las respuestas.

Filtrado de respuestas

Se implementa un filtrado de respuestas basado en el **score de confianza**, eliminando aquellas con un *score* inferior a cierto umbral, por ejemplo, **0.79**. Este enfoque pretende mejorar la calidad de las respuestas al descartar aquellas que el modelo considera menos confiables.

Después de que el modelo BERT ha generado una respuesta para una pregunta dada, se verifica si el *score* de confianza de la respuesta supera cierto umbral. Si el *score* es superior a este umbral, se verifica si se ha encontrado una respuesta válida (no vacía). Si ambos criterios se cumplen, la respuesta se devuelve como la respuesta final. De lo contrario, se devuelve un mensaje indicando que la pregunta debe ser reformulada o que el modelo no está seguro de la respuesta.

Este mecanismo de filtrado basado en el *score* de confianza mejora la calidad de las respuestas proporcionadas por el sistema. Sin embargo, la efectividad de este enfoque depende de la precisión del modelo y puede requerir ajustes finos para optimizar su desempeño.

5.2. Valorar la precisión de las respuestas

Evaluación del rendimiento

Se analiza la exactitud de las respuestas generadas por el asistente en comparación con la información real.

Los resultados de la evaluación se analizan en detalle, considerando los valores obtenidos para las métricas seleccionadas y la calidad general de las respuestas generadas. Se identifican los aspectos que funcionan correctamente y aquellos que requieren mejoras.

Precision: 0.5980392156862745

Recall: 0.13725490196078433

F1-Score: 0.22326797385620914

Accuracy: 0.5686274509803921

Accuracy: La precisión global del modelo, medida por la tasa de respuestas correctas, es 0.568. Esto indica que el modelo tiene una tasa de respuestas correctas significativa en comparación con el total de respuestas, pero aún hay margen de mejora, ya que no es tan alta como se desearía.

Conclusiones

La evaluación del asistente inteligente permite determinar su efectividad en la generación de respuestas precisas y relevantes a las preguntas de los usuarios. Los resultados obtenidos sirven como base para futuras mejoras y optimizaciones del modelo.

En general, estos resultados sugieren que el modelo puede beneficiarse de mejoras en su capacidad para generar respuestas correctas. Es importante revisar el proceso de generación de respuestas, así como la calidad y cantidad de datos utilizados para el entrenamiento, para identificar áreas específicas que requieran ajustes y mejoras.

Recomendaciones

Para mejorar el rendimiento del asistente, hay varias opciones a considerar:

1. *Ampliar el dataset*: Aumentar el tamaño y la diversidad del conjunto de datos puede ayudar al modelo a capturar mejor la variabilidad del lenguaje y mejorar su capacidad para generar respuestas precisas y relevantes.
2. *Afinar los hiperparámetros*: Realizar ajustes en los hiperparámetros del modelo, como la tasa de aprendizaje, el tamaño del lote, la arquitectura del modelo, entre otros, puede ayudar a optimizar el rendimiento del modelo en la tarea específica de generación de respuestas.
3. *Explorar técnicas de preprocesamiento de datos*: Aplicar técnicas avanzadas de preprocesamiento de datos, como la limpieza de texto, la normalización de palabras, la eliminación de ruido y el aumento de datos, puede mejorar la calidad y la consistencia del conjunto de datos, lo que a su vez puede conducir a mejores resultados del modelo.
4. *Experimentar con arquitecturas de modelos alternativos*: Probar diferentes arquitecturas de modelos, como modelos más grandes o modelos específicos diseñados para tareas de generación de texto, podría ayudar a mejorar el rendimiento del modelo en la generación de respuestas.
5. *Exploración de nuevas métricas de evaluación*: Considerar métricas adicionales además de las que ya se están utilizando, como BLEU (*Bilingual Evaluation Understudy*) o ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*).
6. *Implementación de técnicas de filtrado más robustas*: Evaluar la posibilidad de utilizar técnicas de filtrado más avanzadas para mejorar la selección de las respuestas generadas. Por ejemplo, en lugar de simplemente basarse en un umbral de *score* de confianza fijo, considerar métodos más sofisticados que utilicen modelos de puntuación de calidad de respuesta. También se podría explorar el uso de técnicas de filtrado basadas en reglas o aprendizaje automático para identificar y descartar respuestas que sean incoherentes, irrelevantes o incorrectas, lo que podría mejorar la calidad general de las respuestas seleccionadas.

Al implementar estas opciones y realizar experimentos iterativos, se puede mejorar la evaluación del rendimiento del modelo de generación de respuestas y optimizar su capacidad para generar respuestas precisas y relevantes en una variedad de contextos.

5.3. Evaluar y comparar las repuestas con otros LLM [22] [23] [37]



Predeterminado

CohereForAI/c4ai-command-r-plus

Command R+ es el último LLM de Cohere y es el primer modelo de peso abierto que supera a GPT4 en la arena de los chatbots.



mistralai/Mixtral-8x7B-Instruct-v0.1

¡El último modelo MoE de Mistral AI! 8x7B y supera a Llama 2 70B en la mayoría de los benchmarks.



google/gemma-1.1-7b-it

Gemma 7B 1.1 es la última versión de la familia Gemma de modelos ligeros creados por Google, entrenados con un novedoso método RLHF.



NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO

Nous Hermes 2 Mixtral 8x7B DPO es el nuevo modelo insignia de Nous Research entrenado sobre el Mixtral 8x7B MoE LLM.



mistralai/mistral-7B-Instruct-v0.2

Mistral 7B es un nuevo modelo Apache 2.0, lanzado por Mistral AI que supera a Llama2 13B en los puntos de referencia.

Imagen 20: Modelos disponibles en HuggingChat. <https://huggingface.co/chat/models>

Edit Assistant: Mistral-8Bx7B-Instruct-v0.1-Asistente usuarios de la empresa Canal de Isabel II

Modifying an existing assistant will propagate the changes to all users.

Avatar



Delete

Name

Mistral-8Bx7B-Instruct-v0.1-Asistente usuarios de la empresa Canal de

Description

Asistente para los usuarios y clientes de la empresa Canal de Isabel II,
<https://oficinavirtual.canaldeisabelsegunda.es/>

Model

mistralai/Mixtral-8x7B-Instruct-v0.1



Temperature ⓘ

0,9

Top P ⓘ

0,95

Repetition penalty ⓘ

1,1

Top K ⓘ

50

User start messages

Start Message 1

Start Message 2

Start Message 3

Start Message 4

Internet access ⓘ

Experimental

[Give feedback](#)

Default

Assistant will not use internet to do information retrieval and will respond faster.
Recommended for most Assistants.

Web search

Domains search

Specific Links

Instructions (System Prompt)

Asistente para los usuarios y clientes de la empresa Canal
<https://oficinavirtual.canaldeisabelsegunda.es/>

Imagen 21: Configuración de LLM con especificaciones para inferencias.

  <p>Mistral-7B-Instruct-v0.2-Asistente usuarios de la empresa Canal de Isabel II</p> <p>Asistente para los usuarios y clientes de la empresa Canal de Isabel II,...</p> <p>Created by Antonio49</p>	  <p>Mistral-8Bx7B-Instruct-v0.1-Asistente usuarios de la empresa Canal de Isabel II</p> <p>Asistente para los usuarios y clientes de la empresa Canal de Isabel II,...</p> <p>Created by Antonio49</p>
  <p>Google-gemma-Asistente usuarios de la empresa Canal de Isabel II</p> <p>Asistente para los usuarios y clientes de la empresa Canal de Isabel II,...</p> <p>Created by Antonio49</p>	  <p>CohereForAI-Asistente usuarios de la empresa Canal de Isabel II</p> <p>Asistente para los usuarios y clientes de la empresa Canal de Isabel II,...</p> <p>Created by Antonio49</p>
  <p>NousResearch/Mistral-8Bx7B-DPO-Asistente usuarios de la empresa Canal de Isabel II</p> <p>Asistente para los usuarios y clientes de la empresa Canal de Isabel II,...</p> <p>Created by Antonio49</p>	

Imagen 22: **Asistentes**^{BETA}. Creados para evaluar y valorar las respuestas con otros LLM populares.

Asistentes en HuggingChat

<https://huggingface.co/chat/assistants>

Los asistentes son una forma de empaquetar las indicaciones y compartirlas con el mundo. Todos los asistentes son públicos de forma predeterminada con el nombre del autor. La característica más poderosa de los asistentes es que se basan en modelos de código abierto. Así que vamos a probar cada uno de los 5 modelos disponibles para ver cuál se adapta mejor al asistente para usuarios del Canal de Isabel II.

En este estudio, analizamos la capacidad de diferentes modelos de LLM para proporcionar respuestas precisas y relevantes a consultas específicas de los usuarios de la empresa Canal de Isabel II. También evaluamos cómo el fine-tuning de un modelo BERT con un conjunto de datos etiquetados reducido afecta en la longitud y en el contenido de las respuestas generadas.

Configuración de la comparativa

La configuración será común para todos los LLM, para realizar la comparativa. Se utilizan las siguientes especificaciones para todas las inferencias:

- ✚ Nombre: Mistral-7B-Instruct-v0.2-Asistente usuarios de la empresa Canal de Isabel II
- ✚ Instrucciones (mensaje del sistema): Asistente para los usuarios y clientes de la empresa Canal de Isabel II, <https://oficinavirtual.canaldeisabelsegunda.es/>
- ✚ Descripción: Asistente para los usuarios y clientes de la empresa Canal de Isabel II, <https://oficinavirtual.canaldeisabelsegunda.es/>
- ✚ Temperature: 0.9
- ✚ Top P: 0.95
- ✚ Repetition penalty: 1.1
- ✚ Top K: 50
- ✚ Acceso a Internet. Predeterminado: El Asistente no usará Internet para recuperar información y responderá más rápido. Recomendado para la mayoría de los asistentes.
- ✚ Mensaje dinámico: Permite el uso de variables de plantilla `{{url=https://example.com/path}}` para insertar contenido dinámico en su solicitud realizando solicitudes GET a las URL especificadas en cada inferencia.

Modelos de LLM Evaluados

Se evaluaron los siguientes modelos de LLM disponibles en HuggingChat:

- CohereForAI/c4ai-command-r-plus.
- mistralai/Mixtral-8x7B-Instruct-v0.1.
- google/gemma-1.1-7b-it.
- NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO.
- mistralai/mistral-7B-Instruct-v0.2.

Cuestiones utilizadas en las Inferencias

Se utilizaron las siguientes preguntas como base para las inferencias:

- A. ¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?
- B. ¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?

Contexto de referencia

- ✚ "context": "Según establece el Reglamento para el Servicio y Distribución de las Aguas de Canal de Isabel II, (Decreto 2922/1975, de 31 de octubre) podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de la factura, esta no ha sido abonada. El corte del suministro implicaría continuar facturando las cuotas de servicio. El restablecimiento del servicio se realizará una vez liquidada la deuda, así como el importe del restablecimiento. Transcurridos tres meses sin que se haya producido el pago, Canal puede resolver el contrato y proceder a la condena de la acometida (retirada de la instalación). Para volver a tener agua habrá que realizar una nueva contratación."

Respuestas aproximadas:

- A. Podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de la factura, esta no ha sido abonada.
- B. El restablecimiento del servicio se realizará una vez liquidada la deuda, así como el importe del restablecimiento.

Valoración de los Modelos

1. CohereForAI/c4ai-command-r-plus:

- Este modelo merece una valoración positiva, ya que ha demostrado proporcionar respuestas de alta calidad, superando incluso a GPT-4 en la arena de los chatbots.
- Su capacidad para generar respuestas precisas y relevantes lo posiciona como un modelo valioso para aplicaciones de conversación y asistencia al cliente.
- Su desempeño sobresaliente sugiere una mejora significativa en la generación de texto y puede ser una opción preferida para aquellos que buscan la mejor calidad de respuesta.

2. google/gemma-1.1-7b-it:

- Aunque este modelo cuenta con el respaldo de Google, su desempeño en la generación de respuestas en inglés fue extraño y poco satisfactorio.
- La incongruencia en las respuestas en inglés puede indicar limitaciones en su capacidad para comprender y generar contenido en este idioma.
- Dado que el inglés es un idioma ampliamente utilizado en aplicaciones de conversación y asistencia, las respuestas poco claras o extrañas pueden ser un obstáculo significativo para su adopción.

3. NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO:

- Este modelo, siendo la insignia de Nous Research y entrenado sobre el Mixtral 8x7B MoB LLM, merece atención y consideración.
- Su entrenamiento sobre un modelo de alta capacidad como Mixtral 8x7B sugiere que puede ofrecer respuestas de calidad y versatilidad en una variedad de tareas de procesamiento de lenguaje natural.
- Su calidad y desempeño deberían evaluarse en comparación con otros modelos en aplicaciones específicas para determinar su idoneidad.

4. mistralai/Mixtral-8x7B-Instruct-v0.1:

- Este modelo se destaca por su entrenamiento en varios idiomas, incluido el español, lo que lo hace especialmente útil para aplicaciones que requieren soporte en diferentes lenguajes.
- Su capacidad para proporcionar respuestas precisas y relevantes, especialmente en español, lo posiciona como una opción sólida para tareas de asistencia al cliente y generación de contenido multilingüe.

- Su desempeño consistente y confiable lo convierte en una opción valiosa para aplicaciones que requieren comprensión y generación de texto en varios idiomas.

5. mistralai/mistral-7B-Instruct-v0.2:

- Aunque este modelo comparte la misma plataforma que Mixtral-8x7B-Instruct-v0.1, su desempeño varió significativamente.
- Se observaron datos erróneos en las respuestas, como información incorrecta sobre números de teléfono, lo que indica posibles limitaciones en la precisión y la calidad de las respuestas generadas.
- La falta de precisión en las respuestas puede afectar la confiabilidad del modelo en aplicaciones de asistencia al cliente y generación de contenido informativo.

Conclusiones

1. **Diversidad de Modelos:** La disponibilidad de una variedad de modelos de LLM en plataformas como HuggingFace proporciona a los usuarios una amplia gama de opciones para seleccionar el modelo más adecuado para sus necesidades específicas.
2. **Variedad de Respuestas:** Observamos que cada modelo de LLM generó respuestas ligeramente diferentes para las mismas preguntas. Esto destaca la importancia de probar múltiples modelos para obtener una visión más completa de las capacidades de generación de texto.
3. **Desempeño Sobresaliente:** El Modelo CohereForAI/c4ai-command-r-plus demostró un desempeño sobresaliente al proporcionar respuestas precisas y relevantes. Supera incluso a modelos más establecidos como GPT-4 en la arena de los chatbots.
4. **Especialización y Multilingüismo:** Modelos como mistralai/Mixtral-8x7B-Instruct-v0.1 se destacan por su entrenamiento en varios idiomas, lo que los hace especialmente útiles para aplicaciones que requieren soporte multilingüe. Además, la especialización en tareas específicas, como la generación de instrucciones, puede mejorar la precisión y relevancia de las respuestas generadas.

5. **Limitaciones y Errores:** Algunos modelos, como mistralai/mistral-7B-Instruct-v0.2 y NousResearch/Nous-Hermes-2-Mixtral-8x7B-DPO, mostraron limitaciones en la precisión y la calidad de las respuestas generadas, incluyendo datos erróneos en la información proporcionada. Estas limitaciones deben ser consideradas al seleccionar un modelo para aplicaciones críticas donde la precisión es fundamental.
6. **Consideraciones Multifacéticas:** La elección del modelo de LLM adecuado debe basarse en una variedad de factores, incluida la precisión, la relevancia, el multilingüismo, la especialización en tareas específicas y la confiabilidad de las respuestas generadas.
7. **Fine-Tuning de BERT:** Al ajustar finamente un modelo BERT con un conjunto de datos etiquetados reducido, observamos que las respuestas generadas fueron más cortas y directas. Esto puede ser beneficioso en situaciones donde se requiere una respuesta concisa y específica.
8. **Evaluación Individual:** Cada modelo tiene sus propias fortalezas y debilidades, por lo que es importante evaluarlos individualmente en función de los requisitos y objetivos específicos del proyecto. El modelo seleccionado inicialmente, el mistralai/Mixtral-8x7B-Instruct-v0.1, es la opción que mejor resultados puede ofrecernos.

En conclusión, la elección del modelo de LLM y la estrategia de fine-tuning deben basarse en las necesidades del usuario, la naturaleza de las consultas y la disponibilidad de datos etiquetados para entrenamiento. La combinación de varios modelos y técnicas puede proporcionar respuestas más completas y precisas para mejorar la experiencia del usuario.

En el anexo XI, podemos visualizar las imágenes con los resultados obtenidos.

5.4. Identificar áreas de mejora

✚ Para el modelo "**dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa**" se aplicarán estas técnicas para el fine-tuning:

- Fine-tuning supervisado.
- Transfer learning.
- Ajuste de hiperparámetros.
- Data augmentation.

Estas técnicas ayudaron a realizar un fine-tuning efectivo del modelo para la tarea de preguntas y respuestas con tu conjunto de datos específico. Se puede experimentar con diferentes enfoques y ajustes para encontrar la configuración que mejor se adapte a las necesidades y requisitos de rendimiento y además se pueden aplicar otras técnicas como:

- Regularización: Para evitar el sobreajuste en un conjunto de datos pequeño, se puede considerar aplicar técnicas de regularización como la dropout durante el entrenamiento. La dropout puede ayudar a reducir la dependencia entre las neuronas y mejorar la generalización del modelo.
 - Afinación por capas: Dado el tamaño relativamente pequeño del conjunto de datos, se pueden considerar congelar algunas capas del modelo preentrenado y solo ajustar los parámetros de las capas superiores durante el fine-tuning. Esto puede ayudar a evitar el sobreajuste y mejorar la eficiencia del entrenamiento.
- ✚ Ampliar el dataset: Aumentar el tamaño y la diversidad del conjunto de datos puede ayudar al modelo a capturar mejor la variabilidad del lenguaje y mejorar su capacidad para generar respuestas precisas y relevantes.
- ✚ Experimentar con arquitecturas de modelos alternativos: Probar diferentes arquitecturas de modelos, como modelos más grandes o modelos específicos diseñados para tareas de generación de texto, podría ayudar a mejorar el rendimiento del modelo en la generación de respuestas.

6. Resultados y productos obtenidos

- 📌 Notebook de la extracción y preprocesamiento de los datos el texto de una URL.
url = <https://oficinavirtual.canaldeisabelsegunda.es/preguntas-frecuentes>.

<https://colab.research.google.com/drive/1IOBu8NhKAT9h9GgUKaq4oRRIzhGMwnXG>

- 📌 Conjunto de datos en formato JSON que contiene preguntas y respuestas del Canal de Isabel II.

https://drive.google.com/file/d/1wvPeB7L8waVXeEzWHjAGEWdClvW3bgFB/view?usp=drive_link

- 📌 Notebook del proyecto para el Fine-Tuning del modelo BERT pre-entrenado. (Optimizador AdamW).

https://colab.research.google.com/drive/1e2-KS1DW1Xtmg4IU_nPJnl_h2NeARj_o

- 📌 Archivos que conforman el modelo BERT con Fine-Tuning, incluyendo:

<https://github.com/antoniodeepblue/ModeloBERTfine-tuning>

<https://huggingface.co/Antonio49/Personal>

Modelo final: <https://github.com/antoniodeepblue/ModeloCanalBERT>

Modelo final: <https://huggingface.co/Antonio49/ModeloCanal>

- config.json
- model.safetensors
- special_tokens_map.json
- tokenizer.json
- tokenizer_config.json
- training_args.bin
- vocab.txt

🚦 Acceso al interfaz de los asistentes en la plataforma Hugging Face, que incluye:

- Preguntas y respuestas a usuarios utilizando el modelo BERT con Fine-Tuning.

<https://antonio49-bertfinetuningcontextopredefinido2.hf.space>

<https://github.com/antoniodeepblue/CanalBERTfine-tuning>

<https://huggingface.co/spaces/Antonio49/BERTfinetuningContextoPredefinido2>

- Preguntas y respuestas a usuarios utilizando el modelo Mixtral mediante inferencia.

<https://antonio49-chat-mixtral-aguausuarioconhilos.hf.space>

<https://github.com/antoniodeepblue/Chat.Mixtral.AguaUsuarioConHilos>

<https://huggingface.co/spaces/Antonio49/Chat.Mixtral.AguaUsuarioConHilos>

- Asistencia de experto en el sector del agua utilizando el modelo Mixtral mediante inferencia.

<https://antonio49-chat-mixtral-expertoconhilos.hf.space>

<https://github.com/antoniodeepblue/Chat.Mixtral.ExpertoConHilos>

<https://huggingface.co/spaces/Antonio49/Chat.Mixtral.ExpertoConHilos>

🚦 Pruebas de entrenamiento de diferentes versiones:

- Notebook del proyecto utilizando Optimizador Adam junto con LoRA.
- Notebook del proyecto utilizando Optimizador SGD junto con LoRA.
- Notebook que muestra cómo ajustar Mixtral-8x7b en una muestra de ultrachat con QLoRA. Requiere al menos 32 GB de VRAM (al menos 2 GPU de 16 GB si desea utilizar hardware de consumo). En Google Colab, puedes utilizar el A100.

🚦 Carpeta principal del TFG:

<https://drive.google.com/drive/folders/1U3TzPPWYLWy1MWekZ5t0nUU58gJN3HMo>

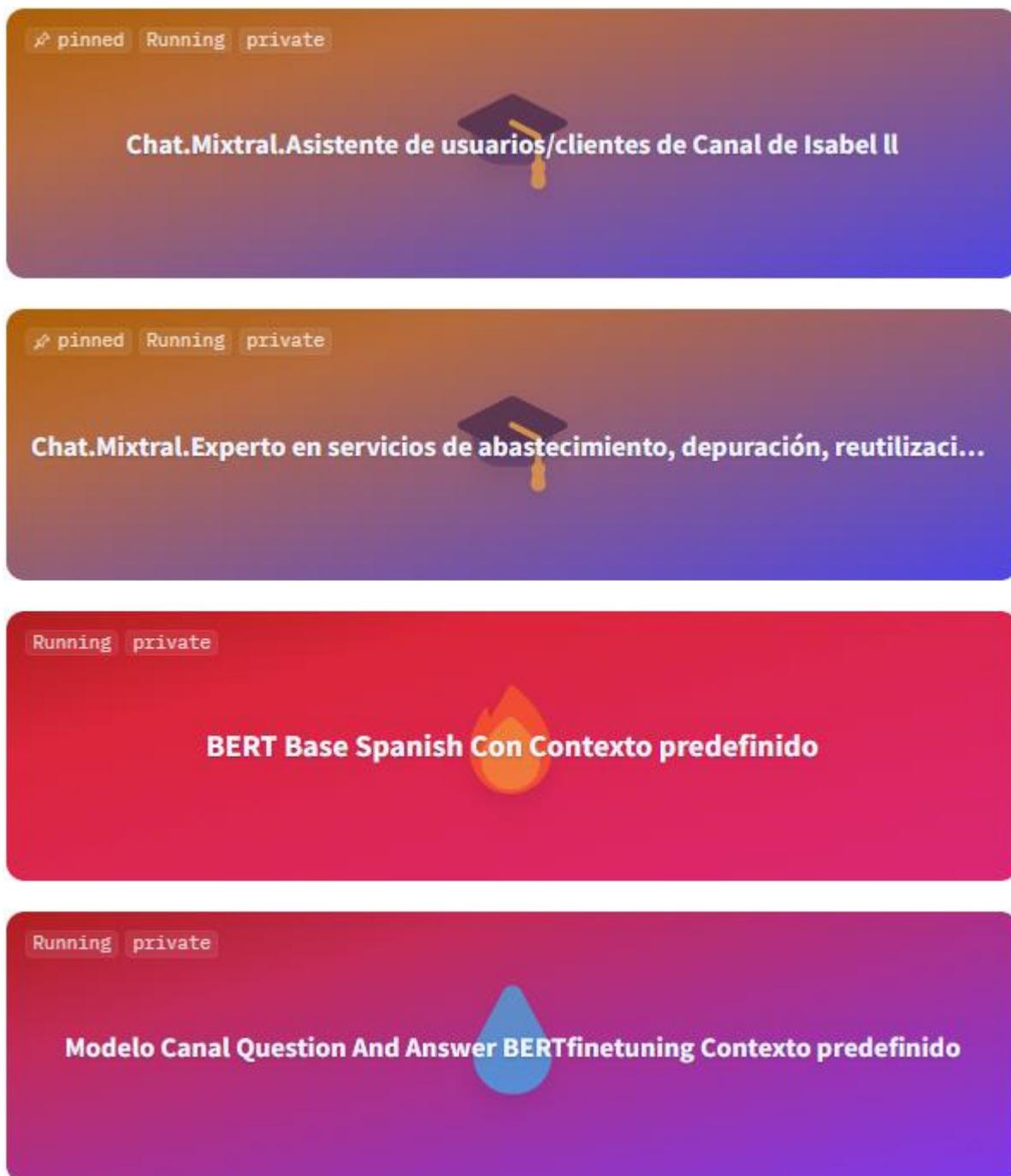


Imagen 23: Espacios creados en Hugging Face. <https://huggingface.co/Antonio49>

7. Conclusiones

Lecciones aprendidas

A lo largo del desarrollo del proyecto se han aprendido diversas lecciones:

- **Importancia de la planificación:** La planificación inicial del proyecto ha sido fundamental para el éxito de este. La distribución eficiente de tareas y la definición de hitos claros han permitido un desarrollo ordenado.
- **Flexibilidad en la metodología:** Aunque se siguió un enfoque en cascada, la incorporación de estrategias como el prototipado y las pruebas tempranas permitió adaptaciones ágiles, facilitando ajustes en objetivos y metodologías conforme surgían resultados.
- **Valor del conjunto de datos:** La calidad y tamaño del conjunto de datos demostraron ser determinantes para el rendimiento del modelo preentrenado. La ampliación gradual del conjunto y su análisis constante contribuyeron a mejorar la precisión del asistente inteligente.
- **Estudio sobre los modelos preentrenados:** La investigación profunda y comparativa de modelos preentrenados en el ámbito del lenguaje natural ha enriquecido la comprensión y selección del modelo más adecuado para el proyecto.

Reflexión crítica sobre el logro de objetivos

En términos generales, se han alcanzado los objetivos principales del proyecto:

- **Investigación sobre modelos preentrenados:** Se ha realizado una revisión de gran alcance de la literatura sobre modelos preentrenados de lenguaje natural, identificando sus características, ventajas y desventajas.
- **Selección y ajuste de modelo preentrenado:** Se optó por el modelo BERT por su desempeño en comprensión del lenguaje natural y la variedad de versiones, y se realizó su ajuste (fine-tuning) al dominio del sector del agua.

- **Selección de modelo para inferencia:** Se ha seleccionado uno de los modelos de Mixtral, que establece el estándar de eficiencia, está disponible de forma gratuita y es adecuado para tareas generales de generación de texto.
- **Desarrollo del asistente inteligente:** Se ha implementado un asistente inteligente utilizando el modelo BERT preentrenado ajustado, incluyendo la interfaz de usuario. Además, de implementar el despliegue del modelo Mixtral en Hugging Face.

Sin embargo, cabe destacar que algunos objetivos específicos se han modificado a lo largo del proyecto en función de los resultados obtenidos:

- **Ampliación del conjunto de datos:** Se incrementó el tamaño del conjunto de datos para mejorar el rendimiento del modelo.
- **Técnicas de fine-tuning:** Se optó por un enfoque de fine-tuning básico debido a las limitaciones de tiempo y recursos. Se recomiendan técnicas de fine-tuning más avanzadas como líneas de trabajo futuro.

Análisis crítico del seguimiento de la planificación y metodología

El proyecto ha seguido en gran medida la planificación inicial, con algunas puntualizaciones:

- **Metodología en cascada:** Se aplicó de manera efectiva, cumpliendo con las etapas secuenciales de análisis, diseño, implementación y evaluación.
- **Investigación continua sobre modelos preentrenados:** La investigación se llevó a cabo a lo largo del proyecto, adaptándose a las necesidades y avances.
- **Recopilación y análisis progresivo del conjunto de datos:** La recopilación y análisis de datos se realizó de forma iterativa, permitiendo ajustes en función de resultados preliminares.

Las modificaciones realizadas han sido necesarias para asegurar el éxito del proyecto y optimizar el desempeño del asistente inteligente.

Líneas de trabajo futuro

El proyecto abre diversas líneas de trabajo futuro que podrían ampliar y mejorar el asistente inteligente:

- **Exploración de técnicas avanzadas de fine-tuning:** Implementar técnicas como la regularización o la afinación (congelación y/o agregación) por capas para mejorar el rendimiento del modelo y evitar el sobreajuste.
- **Ampliación y diversificación del conjunto de datos:** Incorporar más datos de diferentes áreas del sector del agua para mejorar la generalización del modelo.
- **Evaluación y exploración de otros modelos preentrenados:** Investigar modelos adicionales como Mixtral y su aplicabilidad en el contexto del proyecto.
- **Exploración de otras aplicaciones:** Investigar la aplicabilidad del asistente inteligente en otros sectores o ámbitos.
- **Integración con sistemas existentes:** Integrar el asistente inteligente con los sistemas informáticos existentes para una mayor eficiencia y accesibilidad.
- **Evaluación continua y ajustes:** Evaluar el rendimiento del asistente a lo largo del tiempo y realizar ajustes periódicos para mantener su eficacia.

En este trabajo, se ha explorado el potencial de los asistentes basados en modelos de lenguaje pre-entrenados como BERT y Mixtral, y los resultados obtenidos han sido altamente satisfactorios. En mi opinión y tras la evaluación, el modelo Mixtral-8x7B-Instruct-v0.1 se destaca como la opción más idónea para ofrecer respuestas precisas en el contexto del Canal de Isabel II, gracias a su capacidad para interpretar y seguir instrucciones, respaldada por el acceso a una amplia gama de datos pre-entrenados disponibles en la web.

En resumen, el trabajo ha demostrado el potencial transformador de la inteligencia artificial en la mejora de la atención al cliente en el sector del agua. Además, las líneas de trabajo futuro propuestas ofrecen oportunidades para continuar afinando y ampliando las capacidades del asistente inteligente.

8. Glosario

- **Asistente Inteligente:** Un sistema de software que utiliza inteligencia artificial para comprender y responder a las preguntas o solicitudes de los usuarios.
- **BERT:** *Bidirectional Encoder Representations from Transformers*. Es un modelo de lenguaje natural preentrenado desarrollado por Google que utiliza la arquitectura de Transformers.
- **Canal de Isabel II:** La empresa pública española responsable del suministro de agua potable y alcantarillado en la Comunidad de Madrid.
- **Chatbot:** Un programa de computadora que simula una conversación con usuarios humanos a través de texto o voz.
- **Conjunto de Datos:** Una colección de datos estructurados que se utiliza para entrenar o evaluar modelos de aprendizaje automático.
- **Corpus:** Una colección de textos que se utiliza para entrenar o evaluar modelos de lenguaje natural.
- **Data Augmentation (Aumento de Datos):** Una técnica que consiste en generar artificialmente o de forma manual, nuevos datos a partir de datos existentes para aumentar el tamaño y la diversidad del conjunto de datos de entrenamiento. Esto puede ayudar a mejorar el rendimiento del modelo y evitar el sobreajuste.
- **Data collator:** Componente utilizado en el procesamiento de datos que se encarga de combinar y formatear los datos de entrada para su uso en el entrenamiento del modelo.
- **Entrenamiento:** El proceso de ajustar un modelo de aprendizaje automático a un conjunto de datos para mejorar su rendimiento.
- **Extracción de Información:** El proceso de identificar y extraer información relevante de un conjunto de datos.
- **Fine-tuning:** Proceso de ajuste de un modelo preentrenado utilizando datos específicos de dominio para adaptarlo a una tarea específica.

- **Framework:** Estructura o conjunto de reglas que proporciona una base para el desarrollo de aplicaciones, sistemas o soluciones. Pueden incluir componentes específicos para la Interfaz de Usuario (UI).
- **Hiperparámetro:** Un parámetro que controla el comportamiento de un modelo de aprendizaje automático, como el número de capas en una red neuronal o la tasa de aprendizaje.
- **Hugging Face:** Una plataforma online que ofrece acceso a una amplia gama de modelos preentrenados de lenguaje natural y herramientas para su desarrollo.
- **Inferencia:** El proceso de utilizar un modelo preentrenado para generar una salida a partir de una entrada.
- **Learning Rate:** Un parámetro que controla la velocidad a la que se actualizan los pesos de un modelo durante el entrenamiento.
- **Modelo Preentrenado:** Un modelo de aprendizaje automático que ha sido entrenado previamente en una gran cantidad de datos para aprender representaciones generales del lenguaje, y que puede ser ajustado posteriormente para realizar tareas específicas.
- **Mixtral:** Modelo de lenguaje preentrenado de alto rendimiento desarrollado por Mixtral AI, conocido por su eficiencia y calidad en la generación de texto.
- **Overfitting (sobreajuste):** Fenómeno en el que un modelo de aprendizaje automático se ajusta demasiado bien a los datos de entrenamiento y no generaliza bien a datos no vistos.
- **Pipeline (tubería):** Una serie de pasos secuenciales que se realizan para procesar datos o completar una tarea, como el procesamiento del lenguaje natural (NLP) o el entrenamiento de modelos de aprendizaje automático.
- **Procesamiento del lenguaje natural (PNL):** Campo de la inteligencia artificial que se ocupa de la interacción entre las computadoras y el lenguaje humano.
- **Prompt:** Una instrucción o frase que se da a un modelo de lenguaje natural para guiar su respuesta.

- **Prototipado:** Creación de versiones preliminares o prototipos de un producto o sistema para probar su funcionalidad y obtener retroalimentación antes de la implementación completa.
- **Pruebas Tempranas:** El proceso de evaluar un sistema de software en sus primeras etapas de desarrollo para identificar y corregir errores.
- **Regularización:** Técnica utilizada en el entrenamiento de modelos de aprendizaje automático para evitar el sobreajuste, limitando la complejidad del modelo.
- **SGD: Stochastic Gradient Descent.** Es un algoritmo de optimización utilizado para entrenar modelos de aprendizaje automático que actualiza iterativamente los pesos del modelo utilizando el gradiente de la función de pérdida.
- **Técnicas de Fine-tuning:** Diversas estrategias que se pueden aplicar para ajustar un modelo preentrenado a un conjunto de datos específico, como la regularización o la afinación por capas.
- **Token:** Una unidad básica de texto, como una palabra o un subpalabra.
- **Tokenización:** Proceso de dividir un texto en unidades más pequeñas llamadas tokens, que pueden ser palabras, subpalabras o caracteres.
- **Red neuronal:** Modelo computacional inspirado en el funcionamiento del cerebro humano, compuesto por nodos interconectados que procesan y transmiten información.
- **Vectorización:** El proceso de convertir texto en una representación numérica que pueda ser procesada por un modelo de aprendizaje automático.
- **Waterfall (Cascada):** Método de desarrollo de software que sigue una secuencia lineal de etapas, incluyendo requisitos, diseño, implementación, verificación y mantenimiento.
- **Weight Decay (Decaimiento de peso):** En el contexto del entrenamiento de modelos de aprendizaje automático, el *weight decay* es una técnica de regularización que se utiliza para prevenir el sobreajuste al penalizar los valores grandes de los pesos de un modelo durante el proceso de optimización.

9. Fuentes bibliográficas consultadas

- Joaquim Moré López, Introducción al procesamiento del lenguaje natural. PID_00287316, 1ª Edición, Fundació Universitat Oberta de Catalunya (FUOC), Barcelona, 2022. ^[1]
- Smaller BERT Models. [en línea] [consulta: 1 de abril de 2024]. Disponible en: <https://github.com/google-research/bert> ^[2]
- BERT and Transfer Learning in NLP. [en línea] [consulta: 1 de abril de 2024]. Disponible en: <https://medium.com/@mervebdurna/bert-and-transfer-learning-in-nlp-11fc19435fa0> ^[3]
- Fine-Tuning BERT for text classification with LoRA. [en línea] [consulta: 2 de abril de 2024]. Disponible en: <https://medium.com/@karkar.nizar/fine-tuning-bert-for-text-classification-with-lora-f12af7fa95e4> ^[4]
- BERT para responder preguntas sobre SQuAD 2.0 en español. [en línea] [consulta: 3 de abril de 2024]. Disponible en: <https://jaimesendraberenguer.medium.com/bert-para-responder-preguntas-sobre-squad-2-0-en-español-5842748f051a> ^[5]
- SQUAD_es_v8_GPU.ipynb. [en línea] [consulta: 4 de abril de 2024]. Disponible en: https://colab.research.google.com/github/jaisenbe58r/iAApi-QAS-BERT/blob/main/SQUAD_es_GPU.ipynb ^[6]
- Saca el máximo partido de tu suscripción a Colab. [en línea] [consulta: 4 de abril de 2024]. Disponible en: <https://colab.research.google.com/notebooks/pro.ipynb> ^[7]
- Mastering BERT: A Comprehensive Guide from Beginner to Advanced in Natural Language Processing (NLP). [en línea] [consulta: 5 de abril de 2024]. Disponible en: <https://medium.com/@shaikhrayyan123/a-comprehensive-guide-to-understanding-bert-from-beginners-to-advanced-2379699e2b51> ^[8]
- Analyzing Semantic Equivalence of Sentences Using BERT. [en línea] [consulta: 6 de abril de 2024]. Disponible en: <https://www.analyticsvidhya.com/blog/2022/02/analyzing-semantic-equivalence-of-sentences-using-bert/> ^[9]

- Clasificar texto con BERT. [en línea] [consulta: 7 de abril de 2024]. Disponible en: https://www.tensorflow.org/text/tutorials/classify_text_with_bert?hl=es-419 ^[10]
- Fine-tuning a BERT model. [en línea] [consulta: 8 de abril de 2024]. Disponible en: https://www.tensorflow.org/tfmodels/nlp/fine_tune_bert ^[11]
- Recuperación de preguntas y respuestas del codificador de oraciones universal multilingüe. [en línea] [consulta: 9 de abril de 2024]. Disponible en: https://www.tensorflow.org/hub/tutorials/retrieval_with_tf_hub_universal_encoder_qa?hl=es-419 ^[12]
- Ajuste fino con conjuntos de datos personalizados. [en línea] [consulta: 10 de abril de 2024]. Disponible en: https://huggingface.co/transformers/v3.3.1/custom_datasets.html ^[13]
- Ajuste fino de BERT para preguntas y respuestas. [en línea] [consulta: 10 de abril de 2024]. Disponible en: https://huggingface.co/docs/transformers/tasks/question_answering ^[14]
- Ajuste fino de RoBERTa para preguntas y respuestas. [en línea] [consulta: 10 de abril de 2024]. Disponible en: <https://huggingface.co/mrm8488/roberta-base-1B-1-finetuned-squadv1> ^[15]
- QLoRa: Ajuste un modelo de lenguaje grande en su GPU. [en línea] [consulta: 11 de abril de 2024]. Disponible en: <https://kaitechup.substack.com/p/qlora-fine-tune-a-large-language-model-on-your-gpu-27bed5a03e2b> ^[16]
- PEFT (Parameter-Efficient Fine-Tuning). Hugging Face. [en línea] [consulta: 11 de abril de 2024]. Disponible en: <https://huggingface.co/docs/peft/main/en/index> ^[17]
- Ajuste fino del modelo de lenguaje cuantificado usando LoRA con peft y transformadores en GPU T4. [en línea] [consulta: 10 de abril de 2024]. Disponible en: <https://towardsdev.com/fine-tune-quantized-language-model-using-lora-with-peft-transformers-on-t4-gpu-287da2d5d7f1> ^[18]
- QLoRA: ajuste eficiente de LLM cuantificados. [en línea] [consulta: 11 de abril de 2024]. Disponible en: <https://github.com/artidoro/qlora> ^[19]

- LoRA y QLoRA: métodos efectivos para afinar sus LLM en detalle. [en línea] [consulta: 12 de abril de 2024]. Disponible en: <https://medium.com/@levxn/lora-and-qlora-effective-methods-to-fine-tune-your-llms-in-detail-6e56a2a13f3c> [20]
- SomosNLP. [en línea] [consulta: 12 de abril de 2024]. Disponible en: https://somosnlp.org/recursos/tutoriales/07_entrenamiento_finetuning [21]
- Welcome Gemma - Google's new open LLM. [en línea] [consulta: 13 de abril de 2024]. Disponible en: <https://huggingface.co/blog/gemma> [22]
- HuggingChat. [en línea] [consulta: 13 de abril de 2024]. Disponible en: <https://huggingface.co/chat/> [23]
- Hugging Face. About Question Answering. [en línea] [consulta: 14 de abril de 2024]. Disponible en: <https://huggingface.co/tasks/question-answering> [24]
- Canal de Isabel II. [en línea] [consulta: 21 de abril de 2024]. Disponible en: <https://www.canaldeisabelsegunda.es> [25]
- Canal de Isabel II, oficina virtual. [en línea] [consulta: 22 de abril de 2024]. Disponible en: <https://oficinavirtual.canaldeisabelsegunda.es> [26]
- Repositorio de modelos de HuggingFace. [en línea] [consulta: 23 de abril de 2024]. Disponible en: <https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased> [27]
- BETO: Español BERT. [en línea] [consulta: 24 de abril de 2024]. Disponible en: <https://github.com/dccuchile/beto> [28]
- DCC UChile. Departamento de Ciencias de la Computación Universidad de Chile. [en línea] [consulta: 25 de abril de 2024]. Disponible en: <https://github.com/dccuchile> [29]
- Conjunto de datos para el Gran Corpus Español. [en línea] [consulta: 26 de abril de 2024]. Disponible en: https://huggingface.co/datasets/large_spanish_corpus [30]
- Mistral IA generativa abierta y portátil para desarrolladores y empresas. [en línea] [consulta: 30 de abril de 2024]. Disponible en: <https://mistral.ai> [31]
- Mistral mistralai/Mixtral-8x7B-Instruct-v0.1. [en línea] [consulta: 30 de abril de 2024]. Disponible en: <https://huggingface.co/mistralai> [32]

- Mistral Transformer. [en línea] [consulta: 30 de abril de 2024]. Disponible en: <https://github.com/mistralai/mistral-src> [33]
- Mixtral-8x7B: Understanding and Running the Sparse Mixture of Experts by Mistral AI. [en línea] [consulta: 1 de mayo de 2024]. Disponible en: <https://kaiichup.substack.com/p/mixtral-8x7b-understanding-and-running> [34]
- (Parte 1) Construye tu propio RAG con Mistral-7B y LangChain. [en línea] [consulta: 2 de mayo de 2024]. Disponible en: <https://medium.com/@thakermadhav/build-your-own-rag-with-mistral-7b-and-langchain-97d0c92fa146> [35]
- (Parte 2) Construye tu propio RAG con Mistral-7B y LangChain. [en línea] [consulta: 2 de mayo de 2024]. Disponible en: <https://medium.com/@thakermadhav/part-2-build-a-conversational-rag-with-langchain-and-mistral-7b-6a4ebe497185> [36]
- Modelos preentrenados Kaggle. [en línea] [consulta: 3 de mayo de 2024]. Disponible en: <https://www.kaggle.com/models?query=text-embedding&tftub-redirect=true> [37]
- Encontrar el equilibrio adecuado entre el uso de la memoria y la velocidad de inferencia. [en línea] [consulta: 3 de mayo de 2024]. Disponible en: <https://towardsdatascience.com/run-mixtral-8x7b-on-consumer-hardware-with-expert-offloading-bd3ada394688> [38]
- Bibliotecas:
 - 📦 Transformers: <https://github.com/huggingface/transformers>
 - 📦 TensorFlow: <https://www.tensorflow.org/>
 - 📦 PyTorch: <https://pytorch.org/>
- Uso de herramientas de inteligencia artificial generativa (IAG). Modelos preentrenados para diferentes aplicaciones dentro del ámbito del NLP y la generación de texto que se han utilizado para investigar, buscar y contrastar información, conseguir puntos clave, traducir textos, etc. referente al TFG:
 - 📦 ChatGPT (OpenAI). <https://chat.openai.com>
 - 📦 Mixtral. <https://mistral.ai>
 - 📦 Gemini (Google). <https://gemini.google.com>
 - 📦 Copilot. <https://www.microsoft.com/es-es/edge/features/copilot?form=MT00D8>

10. Anexos

10.1. Anexo I: Diagrama Gantt: Tareas del Proyecto.

Nombre de tarea	Duraz	Comienzo	Fin	Pr
▷ Gestión del proyecto TFG.	89 días	mié 28/02/24	dom 30/06/24	
Entrega PECO	0 días	lun 11/03/24	lun 11/03/24	2
◀ Desarrollo tareas del TFG.	70 días	mar 12/03/24	dom 16/06/24	
TFG 1. • Elaborar Plan de Trabajo.	5 días	mié 13/03/24	mar 19/03/24	3
TFG 2. • Desarrollar Diagrama de Gantt. • Redactar primera parte de la memoria del trabajo.	4 días	mié 20/03/24	lun 25/03/24	5
Entrega PEC1 - Investigar la aplicabilidad de modelos preentrenados.	0 días	mar 26/03/24	mar 26/03/24	6
TFG 3. • Revisar la literatura existente sobre el uso de modelos preentrenados.	6 días	mié 27/03/24	mié 03/04/24	7
TFG 4. • Análisis de modelos preentrenados para fine-tuning.	5 días	jue 04/04/24	mié 10/04/24	8
TFG 5. • Recopilación y análisis del conjunto de datos.	5 días	jue 11/04/24	mié 17/04/24	9
TFG 6. • Seleccionar modelos preentrenados adecuados para fine-tuning.	3 días	jue 18/04/24	lun 22/04/24	10
TFG 7. • Definir los parámetros y características específicas del fine-tuning para adaptar el modelo al contexto del Canal de Isabel II.	3 días	mar 23/04/24	jue 25/04/24	11
Realizar fine-tuning e inferencia en modelos preentrenados	0 días	vie 26/04/24	vie 26/04/24	12
TFG 8. • Seleccionar un modelo preentrenado adecuado para ajustar parámetros de generación de texto o realizar tareas de PNL para una inferencia específica en el contexto del Canal de Isabel II.	5 días	vie 26/04/24	jue 02/05/24	13
Entrega PEC2 - Desarrollo del Asistente	0 días	vie 03/05/24	vie 03/05/24	14
TFG 9. • Implementar el asistente inteligente utilizando un modelo preentrenado ajustado.	9 días	vie 03/05/24	mié 15/05/24	15
TFG 10. • Diseñar la interfaz del asistente para facilitar la interacción de los clientes o técnicos del Canal de Isabel II.	5 días	jue 16/05/24	mié 22/05/24	16
TFG 11. • Incorporar capacidades de ofrecer soluciones técnicas especializadas a consultas específicas del sector de agua.	4 días	jue 23/05/24	mar 28/05/24	17
Entrega PEC3 - Evaluación del Asistente.	0 días	mié 29/05/24	mié 29/05/24	18
TFG 12. • Recopilación de datos para la evaluación del asistente.	2 días	mié 29/05/24	jue 30/05/24	19
TFG 13. • Valorar la precisión de las respuestas.	2 días	sáb 01/06/24	lun 03/06/24	20
TFG 14. • Evaluar y comparar las respuestas con otros LLM.	2 días	mar 04/06/24	mié 05/06/24	21
TFG 15. • Identificar áreas de mejora.	2 días	jue 06/06/24	vie 07/06/24	22
Resultados y productos obtenidos	0 días	sáb 08/06/24	sáb 08/06/24	
TFG 16. • Despliegue de los asistente en la nube con acceso a cualquier usuario.	6 días	sáb 08/06/24	vie 14/06/24	22
Entrega PEC4	0 días	dom 16/06/24	dom 16/06/24	25

Imagen 24: Diagrama Gantt. Tareas del Proyecto.

Nombre de tarea	Dur:	Comienzo	Fin	Pr
UOC	89 días	mié 28/02/24	dom 30/06/24	
PEC0: Redactar la propuesta del TFG.	9 días	mié 28/02/24	dom 10/03/24	
Entrega de la PEC0.	0 días	lun 11/03/24	lun 11/03/24	
PEC1 :	11 días	mar 12/03/24	mar 26/03/24	
Descripción del TFG.	4 días	mar 12/03/24	vie 15/03/24	28
Objetivos generales y específicos.	2 días	lun 18/03/24	mar 19/03/24	31
Planificación con hitos y temporización con un diagrama de Gantt.	4 días	mié 20/03/24	lun 25/03/24	32
Entrega de la PEC1.	0 días	mar 26/03/24	mar 26/03/24	33
PEC2:	28 días	mié 27/03/24	vie 03/05/24	
Informe de seguimiento del proyecto.	10 días	mar 26/03/24	lun 08/04/24	34
Revisión del plan de trabajo.	11 días	mié 10/04/24	mié 24/04/24	36
Redacción y revisión de entregables parciales.	6 días	jue 25/04/24	jue 02/05/24	37
Redacción y entrega de la PEC2.	0 días	vie 03/05/24	vie 03/05/24	38
PEC3:	19 días	sáb 04/05/24	mié 29/05/24	
Informe de seguimiento del proyecto.	6 días	sáb 04/05/24	vie 10/05/24	39
Revisión del plan de trabajo.	6 días	mar 14/05/24	mar 21/05/24	41
Redacción y revisión de entregables parciales.	5 días	mié 22/05/24	mar 28/05/24	42
Redacción y entrega de la PEC3.	0 días	mié 29/05/24	mié 29/05/24	43
PEC4:	13 días	jue 30/05/24	dom 16/06/24	
Recopilación y revisión de documentos.	6 días	jue 30/05/24	jue 06/06/24	44
Redacción de la memoria del trabajo final.	7 días	vie 07/06/24	sáb 15/06/24	46
Entrega de PEC4.	0 días	dom 16/06/24	dom 16/06/24	47
PEC5a:	6 días	lun 17/06/24	dom 23/06/24	
Elaboración de la presentación del TFG.	6 días	lun 17/06/24	sáb 22/06/24	48
Entrega de la PEC5a.	0 días	dom 23/06/24	dom 23/06/24	50
PEC5b:	6 días	lun 24/06/24	dom 30/06/24	
Elaborar informe de autoevaluación.	5 días	lun 24/06/24	vie 28/06/24	51
Defensa en el foro.	5 días	lun 24/06/24	vie 28/06/24	51
Entrega de la PEC5b.	0 días	dom 30/06/24	dom 30/06/24	53

Imagen 25: Diagrama Gantt. Tareas del Proyecto.

10.2. Anexo II: Tabla de hitos parciales

Actividad	Inicio	Entrega
PEC0	28/02/2024	11/03/2024
PEC1	12/03/2024	26/03/2024
PEC2	27/03/2024	03/05/2024
PEC3	04/05/2024	29/05/2024
PEC4	30/05/2024	16/06/2024
PEC5a	17/06/2024	23/06/2024
PEC5b	24/06/2024	30/06/2024

10.3. Anexo III: BERT en español [2] [3] [5] [8] [9]

¿Qué es BERT?

- BERT es una biblioteca de código abierto creada por Google en 2018.
- Su nombre completo es **Representaciones de codificador bidireccional de Transformer**.
- A diferencia de otras técnicas que analizan oraciones de izquierda a derecha o de derecha a izquierda, BERT utiliza el codificador Transformer en ambas direcciones.
- Su objetivo es generar un modelo de lenguaje que comprenda contextos complejos y relaciones entre palabras.
- BERT se ha utilizado con éxito en diversas tareas de **Procesamiento del Lenguaje Natural (NLP)**, como predicción de intenciones, respuesta a preguntas y clasificación de texto.

Ventajas de BERT para Fine-Tuning:

- **Reducción de costos de computación:** No necesitas entrenar un modelo desde cero, lo que ahorra tiempo y recursos.
- **Huella de carbono reducida:** Al aprovechar un modelo preentrenado, contribuyes menos al consumo energético.
- **Acceso a modelos de última generación:** Puedes utilizar modelos avanzados sin la necesidad de entrenarlos completamente.

Modelos preentrenados de BERT en español:

- **BETO (Bidirectional Encoder Representations from Transformers for Spanish):** Es una versión de BERT preentrenada específicamente para el idioma español. Ha demostrado buen rendimiento en una variedad de tareas del procesamiento del lenguaje natural, incluyendo preguntas y respuestas.

- **CamemBERT:** Aunque inicialmente se entrenó para el francés, también se ha adaptado para el español. Al igual que BERT, es una versión de BERT preentrenada específicamente para el español y ha mostrado buenos resultados en tareas de preguntas y respuestas.
- **DistilBERT en español:** Aunque DistilBERT es una versión más ligera y comprimida de BERT, también existen versiones preentrenadas específicamente para el español. A pesar de su menor tamaño, ha demostrado ser útil en tareas de preguntas y respuestas, especialmente en escenarios donde se necesita una inferencia más rápida.

Fine-Tuning con BERT:

- **Preparación:** Clonar el repositorio de BERT y configurar el entorno.
- **Dataset:** Preparar el dataset con títulos, contexto, preguntas y respuestas.
- **Fine-Tuning:** Ajustar el modelo preentrenado de BERT utilizando el dataset específico.
- **Evaluación:** Evaluar el rendimiento del modelo ajustado en tareas como clasificación de texto o respuesta a preguntas.

Datos etiquetados para obtener buenos resultados:

1. Tamaño del Dataset Etiquetado:

- En general, tener más datos etiquetados suele ser beneficioso para el fine-tuning de modelos de lenguaje como BERT. Se ha demostrado que estos modelos pueden beneficiarse significativamente de grandes cantidades de datos de entrenamiento.
- Sin embargo, la mejora en el rendimiento puede disminuir a medida que se aumenta la cantidad de datos, lo que se conoce como el "efecto de saturación".
- Comenzar con un conjunto de datos etiquetados de tamaño moderado y evaluar el rendimiento del modelo. Luego, se considera si necesitas más datos para mejorar aún más.

2. Diversidad y Representatividad:

- La diversidad en los datos etiquetados es importante. Es útil tener datos que cubran una amplia gama de casos y escenarios relevantes para la tarea en cuestión. Esto puede ayudar al modelo a generalizar mejor y a realizar predicciones precisas en situaciones nuevas.
- Representar adecuadamente las clases o categorías que deseas predecir. Si tienes clases desequilibradas, hay que asegurarse de tener suficientes ejemplos de cada clase, aunque el balance de clases es una tarea de clasificación y para este proyecto no debe influir.

3. Transferencia de Aprendizaje:

- BERT se beneficia especialmente de la transferencia de aprendizaje. Utilizar un modelo preentrenado (como BERT) y ajustar los pesos en función de tus datos etiquetados.
- Incluso con un conjunto de datos relativamente pequeño, se pueden obtener buenos resultados si el modelo preentrenado ya ha capturado características generales del lenguaje.

4. Curva de Aprendizaje:

- Observar la curva de aprendizaje a medida que se aumenta la cantidad de datos etiquetados. Si el rendimiento sigue mejorando, continuamos agregando más datos.
- Si la curva se aplana, se considera detener el aumento de datos o explorar otras estrategias, como ajustar hiperparámetros o mejorar la arquitectura del modelo.

5. Aumento de Datos (Data Augmentation):

- Si no tenemos acceso a una gran cantidad de datos etiquetados, se considera aplicar técnicas de aumento de datos. Estas técnicas generan variantes sintéticas de los ejemplos existentes.
- El aumento de datos puede ayudar a diversificar el conjunto de entrenamiento sin necesidad de recopilar más datos reales.

En resumen, no hay una cantidad exacta de datos etiquetados que garantice buenos resultados en todos los casos. Es recomendable experimentar con diferentes tamaños de conjunto de datos y monitorear el rendimiento del modelo en un conjunto de validación para determinar la cantidad óptima de datos necesaria para la tarea específica.

10.4. Anexo IV: Mixtral: Comprensión y funcionamiento ^{[31] [32] [33] [34] [38]}

Mixtral-8x7B es una mezcla dispersa de 8 modelos expertos. En total, contiene 46,7 B (mil millones) de parámetros y ocupa 96,8 GB en el disco duro.

Sin embargo, gracias a esta arquitectura, Mixtral-8x7B puede ejecutarse de manera eficiente en hardware de consumo. La inferencia con Mixtral-8x7B es, de hecho, significativamente más rápida que la de otros modelos de tamaño similar, mientras que los supera en la mayoría de las tareas.

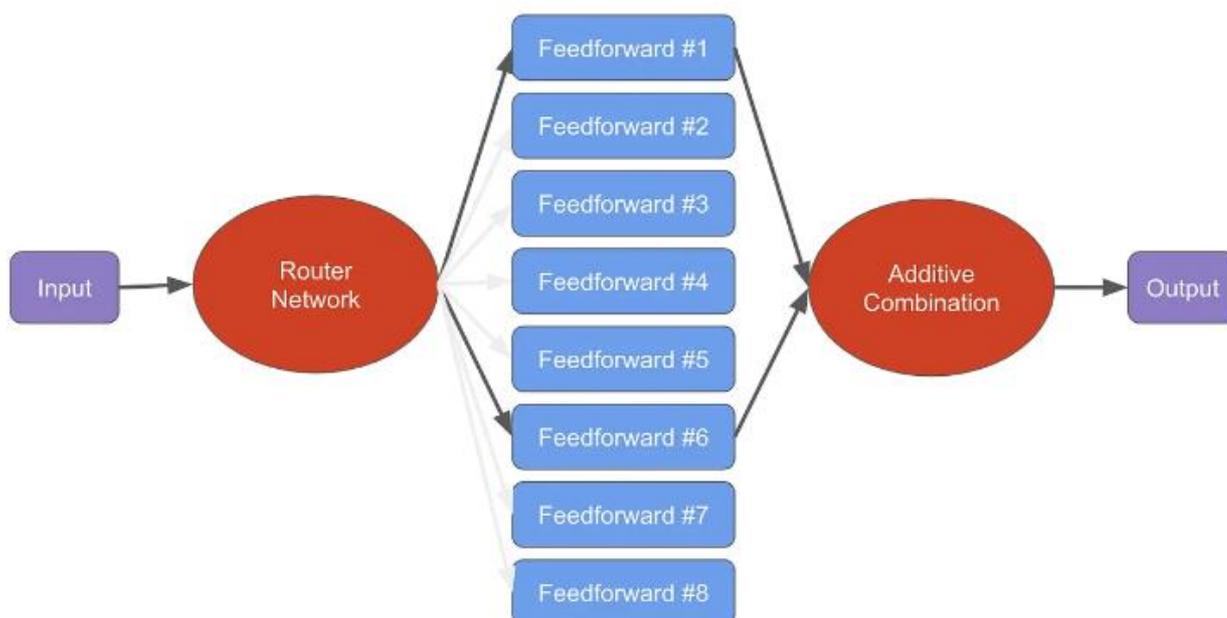


Imagen26: En este ejemplo, la red del router selecciona los expertos #1 y #6 de la arquitectura mixtral para el paso de reenvío. <https://kaitchup.substack.com/p/mixtral-8x7b-understanding-and-running>

Una mezcla dispersa de expertos (*Sparse Mixture of Experts - SMOE*) es un tipo de arquitectura de red neuronal diseñada para mejorar la eficiencia y la escalabilidad de los modelos tradicionales. Se introdujo el concepto de una mezcla de expertos para permitir que un modelo aprenda diferentes partes del espacio de entrada utilizando subredes "expertas" especializadas. En Mixtral, hay 8 subredes de expertos.

Hay que tener en cuenta que en el nombre del modelo "8x7B", se crea cierta confusión. El modelo tiene un total de 46,7 mil millones de parámetros, que son casi 10 mil millones de parámetros menos de lo que producirían los parámetros 8x7B. De hecho, Mixtral-8x7b no es un modelo de parámetros 56B, ya que varios módulos, como los de autoatención, se comparten con las 8 subredes expertas.

En Google Colab, solo funciona con la GPU A100 (necesita Google Colab Pro). El modelo tiene 46.7B parámetros. Dado que los parámetros son bfloat16, necesitamos 2 bytes de memoria por parámetro, es decir, casi 100 GB libres en su disco duro para descargar el modelo.

El modelo está disponible en el formato safetensors. Se divide en 20 fragmentos de alrededor de 4,95 GB. Una vez descargado, puede tardar hasta 7 minutos en cargar el modelo en la GPU.

Cuando se cuantifica a 4 bits, el modelo ocupa 23 GB de VRAM. Para ejecutarlo sin problemas se necesitan al menos dos GPU con 16 GB de VRAM cada una.

El modelo Mixtral es un modelo de lenguaje que combina varias técnicas y componentes para mejorar su rendimiento en tareas de generación de texto. Una explicación detallada de su estructura:

1. Embedding Layer:

- **Embedding (32000, 4096):** Esta capa de embedding convierte los tokens de entrada en vectores densos de tamaño 4096. El tamaño del vocabulario es de 32000 tokens.

2. Capas de Decodificador Mixtral:

- **ModuleList:** Es una lista de capas de decodificador, cada una de las cuales es una instancia de **MixtralDecoderLayer**.
- **32 x MixtralDecoderLayer:** Hay un total de 32 capas de decodificador. Cada una consiste en:
 - **Atención propia de Mixtral ((self_attn): MixtralAttention):** Esta atención propia utiliza proyecciones lineales 4-bit para calcular las consultas, claves y valores. Luego, se aplica una proyección lineal para obtener el resultado final. Además, utiliza una incrustación rotativa MixtralRotaryEmbedding. Una explicación más detallada:

✚ MixtralAttention:

- a. Esta es la capa de atención propia (self-attention) dentro de una capa del decodificador Mixtral.
- b. La atención propia se utiliza para calcular las interacciones entre las diferentes posiciones de los tokens en una secuencia.
- c. La capa MixtralAttention toma como entrada consultas (q), claves (k) y valores (v) y produce una salida ponderada que representa la atención.

✚ Proyecciones Lineales:

- a. **q_proj, k_proj, v_proj, o_proj**: Son proyecciones lineales que transforman las consultas, claves y valores en vectores de diferentes dimensiones.
- b. **Linear4bit(in_features, out_features, bias=False)**: Estas proyecciones lineales son especificadas con precisión de 4 bits, lo que significa que utilizan una cuantización para reducir el uso de memoria y mejorar la eficiencia computacional.
- c. En este caso, las proyecciones lineales tienen las siguientes características:
 - i. **q_proj** y **o_proj** tienen tanto la entrada como la salida de tamaño 4096, lo que significa que no hay cambio de dimensión en las consultas y las salidas.
 - ii. **k_proj** y **v_proj** tienen una dimensión de salida más baja de 1024, lo que puede reducir la complejidad computacional al calcular las puntuaciones de atención.

✚ MixtralRotaryEmbedding:

- a. Esta es una capa que proporciona incrustaciones rotativas, que son una forma de codificar información posicional en la atención propia.
- b. Las incrustaciones rotativas ayudan al modelo a capturar mejor las relaciones entre las diferentes posiciones en la secuencia, sin depender únicamente de la posición absoluta.

En resumen, esta parte del código define la capa de atención propia dentro de una capa del decodificador Mixtral, que utiliza proyecciones lineales con precisión de 4 bits para transformar las consultas, claves y valores, y también utiliza incrustaciones rotativas para mejorar la captura de información posicional.

- **Bloque MoE disperso de Mixtral ((*block_sparse_moe*): *MixtralSparseMoeBlock*):** Este bloque utiliza un enfoque de mezcla de expertos dispersos (MoE) para mejorar la representación del modelo. Consiste en una capa de puerta lineal seguida de ocho expertos, cada uno de los cuales es una instancia de **MixtralBlockSparseTop2MLP**. Una explicación más detallada:

- ✚ **MixtralSparseMoeBlock:**

- a. Esta es una estructura de bloque que implementa el enfoque de mezcla de expertos dispersos (MoE) dentro del decodificador Mixtral.
- b. El objetivo del bloque MoE es mejorar la capacidad del modelo para aprender representaciones más complejas y capturar relaciones más sofisticadas en los datos.

- ✚ **Puerta (gate):**

- a. **(gate): Linear4bit(in_features=4096, out_features=8, bias=False):** Esta es una capa lineal que actúa como una puerta para seleccionar qué expertos se activan en función de la entrada.
- b. La entrada a esta capa es de tamaño 4096, que es la dimensión de la salida de la capa de atención propia.
- c. La salida de esta capa tiene un tamaño de 8, lo que indica que hay 8 expertos en total dentro del bloque MoE.

- ✚ **Lista de Expertos:**

- a. **(experts): ModuleList:** Es una lista de módulos que representan los expertos dentro del bloque MoE.
- b. **(0-7): 8 x MixtralBlockSparseTop2MLP:** Hay un total de 8 expertos en esta lista.
- c. Cada experto es una instancia de **MixtralBlockSparseTop2MLP**, que es un tipo específico de red neuronal multicapa (MLP).

- d. La estructura de cada experto consiste en tres capas lineales, **w1**, **w2** y **w3**, seguidas de una función de activación SiLU (sigmoid-weighted linear unit).

✚ Capas Lineales:

- a. **Linear4bit(in_features, out_features, bias=False):** Cada capa lineal dentro de los expertos utiliza una precisión de 4 bits para reducir el uso de memoria y mejorar la eficiencia computacional.
- b. Las dimensiones de las capas lineales están diseñadas específicamente para la arquitectura del bloque MoE.
- c. En particular, las dimensiones de entrada y salida varían entre las capas **w1**, **w2** y **w3**, lo que permite al experto aprender diferentes representaciones de los datos en cada capa.

En resumen, esta parte del código define un bloque MoE disperso dentro del decodificador Mixtral, que utiliza una puerta lineal para seleccionar expertos y una lista de expertos que consisten en redes neuronales multicapa (MLP) con precisión de 4 bits para mejorar la capacidad del modelo para aprender representaciones más complejas de los datos.

- **Normalización RMS de Mixtral:** La normalización de raíz media cuadrática (RMSNorm) se aplica tanto antes como después de la atención para estabilizar el entrenamiento y mejorar el rendimiento.

3. Capa de Normalización Final:

- **MixtralRMSNorm():** Esta capa de normalización RMS se aplica al final de todas las capas del decodificador.

4. Cabeza de LM (Modelo de Lenguaje):

- **Linear(in_features=4096, out_features=32000, bias=False):** La capa de cabeza del modelo de lenguaje es una capa lineal que mapea las representaciones de los tokens de salida de tamaño 4096 a un espacio de salida de tamaño 32000 (tamaño del vocabulario), sin sesgo (bias).

Se carga el modelo con Transformers para visualizar la estructura del modelo que es más fácil de entender:

```

[ ] print(model)

MixtralForCausalLM(
  (model): MixtralModel(
    (embed_tokens): Embedding(32000, 4096)
    (layers): ModuleList(
      (0-31): 32 x MixtralDecoderLayer(
        (self_attn): MixtralAttention(
          (q_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear4bit(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear4bit(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): MixtralRotaryEmbedding()
        )
        (block_sparse_moe): MixtralSparseMoeBlock(
          (gate): Linear4bit(in_features=4096, out_features=8, bias=False)
          (experts): ModuleList(
            (0-7): 8 x MixtralBlockSparseTop2MLP(
              (w1): Linear4bit(in_features=4096, out_features=14336, bias=False)
              (w2): Linear4bit(in_features=14336, out_features=4096, bias=False)
              (w3): Linear4bit(in_features=4096, out_features=14336, bias=False)
              (act_fn): SiLU()
            )
          )
        )
        (input_layernorm): MixtralRMSNorm()
        (post_attention_layernorm): MixtralRMSNorm()
      )
    )
    (norm): MixtralRMSNorm()
  )
  (lm_head): Linear(in_features=4096, out_features=32000, bias=False)
)

```

Imagen 27: Estructura del modelo Mixtral-8x7B.

Aunque los SMOE son más rápidos que los modelos estándar de tamaño similar siguen ocupando la misma cantidad de memoria. La **cuantización** es una forma de reducir el consumo de memoria. Descargar algunos expertos a una memoria más lenta, por ejemplo, la RAM de la CPU, también puede ser una buena alternativa cuando sabemos de antemano qué experto será solicitado para una tarea determinada.

En conclusión, cada experto es responsable de manejar una región o aspecto específico de los datos de entrada, y una red de compuerta (o enrutador) determina cuánto contribuye cada experto a la predicción final. En el caso de Mixtral, **solo se activan 2 expertos al mismo tiempo**. Se utilizan alrededor de 13B parámetros durante la inferencia, de ahí la inferencia más eficiente en comparación con otros modelos de tamaño similar.

Esta activación dispersa puede ser beneficiosa por varias razones:

- **Eficiencia computacional:** Activar solo un subconjunto de expertos reduce el costo computacional de evaluar todo el grupo de expertos para cada entrada.
- **Eficiencia de los parámetros:** Al permitir que solo unos pocos expertos estén activos para una entrada determinada, el modelo puede asignar sus parámetros de manera más eficiente.
- **Generalización:** La activación dispersa puede alentar al modelo a aprender características más especializadas y detalladas para diferentes regiones del espacio de entrada. Esto puede conducir a una mejor generalización y rendimiento en diversas entradas. Por ejemplo, Mistral AI afirma que Mixtral ha sido entrenado en italiano, alemán, francés, español y también para codificar. Podemos imaginar que pueden haber entrenado a algunos de los 8 expertos para que sean especialmente mejores para estos lenguajes y tareas.

En resumen, el modelo Mixtral utiliza atención propia, bloques MoE dispersos y normalización RMS para mejorar la calidad de las representaciones de los tokens y la capacidad de generación de texto.

Ajustar Mixtral-8x7b en una muestra de ultrachat con QLoRA.

Requiere al menos 32 GB de VRAM (al menos 2 GPU de 16 GB si desea utilizar hardware de consumo). En Google Colab, ejecutando el A100, vemos que se utilizaron: 6,6 GB RAM del sistema, 21.1 GB RAM de la GPU y 118 GB de Disco.



Imagen 28: Fine_tune_Mixtral_8x7B_en_Computadora_(QLoRA). Ejecución.

<https://docs.google.com/document/d/1QS-06nPwlgSYANPBavdrHZKaAoIDJNvKMp74G84odEE/edit>

10.5. Anexo V: Aceleradores por Hardware en Google Colab ^[6] ^[7] ^[16] ^[18]

Google Colab ofrece una variedad de opciones de **aceleradores por hardware** para ejecutar código en la nube. Estos aceleradores proporcionan diferentes niveles de rendimiento y son adecuados para diferentes tipos de tareas. Se detallan las diferencias y utilidades de las opciones disponibles:

Opciones de acelerador:

1. CPU:

- **Descripción:** La CPU (Unidad Central de Procesamiento) es el procesador general del sistema.
- **Ventajas:**
 - Amplia compatibilidad con software.
 - Bajo consumo de energía.
 - Sin costo adicional.
- **Desventajas:**
 - Rendimiento relativamente bajo para tareas computacionalmente intensivas.
 - No ideal para entrenamiento de modelos de aprendizaje automático o procesamiento de grandes conjuntos de datos.

2. T4 GPU:

- **Descripción:** La TPU (Tensor Processing Unit) es un procesador especializado para el aprendizaje automático. La T4 GPU es una versión más pequeña y menos potente de la TPUv4.
- **Ventajas:**
 - Rendimiento significativamente superior a la CPU para tareas de aprendizaje automático y procesamiento de datos.
 - Adecuado para entrenamiento de modelos de tamaño mediano.
- **Desventajas:**
 - Menor rendimiento que las opciones de GPU y TPU más potentes.
 - No ideal para tareas que requieren un rendimiento máximo.

3. TPUv4:

- **Descripción:** La TPUv4 es la última generación de TPU de Google, diseñada para el aprendizaje automático a gran escala.
- **Ventajas:**
 - Rendimiento excepcional para tareas de aprendizaje automático y procesamiento de datos.
 - Ideal para entrenamiento de modelos de gran tamaño y conjuntos de datos masivos.
- **Desventajas:**
 - Mayor consumo de energía que otras opciones.
 - Disponibilidad limitada en comparación con la CPU y la T4 GPU.

4. A100 GPU:

- **Descripción:** La A100 GPU es una GPU de alto rendimiento de NVIDIA, diseñada para computación general y aprendizaje automático.
- **Ventajas:**
 - Rendimiento superior a la T4 GPU y comparable a la TPUv4 para ciertas tareas.
 - Adecuado para una amplia gama de aplicaciones, incluyendo renderizado 3D, simulaciones y aprendizaje automático.
- **Desventajas:**
 - Menor eficiencia energética que la TPUv4.
 - Disponibilidad limitada en comparación con la CPU y la T4 GPU.

5. V100 GPU:

- **Descripción:** La V100 GPU es una GPU de alto rendimiento de NVIDIA, predecesora de la A100 GPU.
- **Ventajas:**
 - Rendimiento similar a la A100 GPU para algunas tareas.
 - Puede ser una opción más económica en algunos casos.

- **Desventajas:**

- Menor eficiencia energética que la TPUv4.
- Menor rendimiento que la A100 GPU para tareas modernas de aprendizaje automático.
- Disponibilidad limitada en comparación con la CPU, la T4 GPU y la A100 GPU.

Selección del acelerador adecuado:

La elección del acelerador adecuado depende de varios factores:

- **Tipo de tarea:**

- **CPU:** Adecuada para tareas básicas, programación general y análisis de datos simples.
- **T4 GPU:** Ideal para aprendizaje automático de tamaño mediano, entrenamiento de redes neuronales y procesamiento de imágenes.
- **TPUv4:** Perfecta para aprendizaje automático a gran escala, entrenamiento de modelos complejos y conjuntos de datos masivos.
- **A100 GPU:** Versátil para computación general, aprendizaje automático, renderizado 3D y simulaciones.
- **V100 GPU:** Opción económica para tareas de aprendizaje automático que no requieren el máximo rendimiento.

- **Tamaño del modelo y del conjunto de datos:**

- Cuanto mayor sea el modelo o el conjunto de datos, mayor será la necesidad de un acelerador potente.

- **Presupuesto:**

- La TPUv4 y la A100 GPU tienen un costo por hora.

- **Disponibilidad:**

- La disponibilidad de las opciones de GPU puede ser limitada en algunos casos.

Tabla comparativa de hardware en aceleradores de Google Colab:

Acelerador	CPU	T4 GPU	TPUv4	A100 GPU	V100 GPU
CPU	Intel Xeon E5-2699 v4	-	-	-	-
RAM del sistema	64 GB	-	-	-	-
RAM de GPU	-	16 GB	-	40 GB	16 GB
Núcleos de CPU	22	-	-	-	-
Núcleos de GPU	-	256	-	64	64
Memoria caché L3	40 MB	-	-	-	-
Tipo de memoria	DDR4	HBM2	HBM2	HBM2	HBM2
Ancho de banda de memoria	64 GB/s	224 GB/s	600 GB/s	672 GB/s	900 GB/s

Imagen 29: Comparativa de hardware en aceleradores de Google Colab.

Fine-tuning de grandes Modelos de Lenguaje (LLM) en Google Colab:

Consideraciones para la elección del acelerador:

Para realizar fine-tuning de grandes modelos de lenguaje (LLM) como **BERT o Mixtral** en Google Colab, la elección del **acelerador de hardware** adecuado es crucial para obtener un buen rendimiento y eficiencia. Algunos factores que considerar son:

1. Tamaño del modelo:
 - Modelos más grandes como **Mixtral** requieren un acelerador más potente.
2. Tamaño del conjunto de datos:
 - Conjuntos de datos más grandes demandan mayor capacidad de procesamiento.
3. Presupuesto:
 - Algunas opciones de GPU tienen un costo por hora después del período gratuito.
4. Disponibilidad:
 - La disponibilidad de las opciones de GPU puede ser limitada en algunos casos.

Recomendaciones de aceleradores:

1. TPUv4:

- **Opción ideal:** La TPUv4 ofrece el mejor rendimiento para fine-tuning de LLMs, con 110 TFLOPS en FP32 y 440 TFLOPS en FP16.
- **Alta eficiencia energética:** Consume menos energía que las GPUs.

2. A100 GPU:

- **Buena alternativa:** La A100 GPU ofrece un rendimiento considerable con 90 TFLOPS en FP32 y 180 TFLOPS en FP16. (Mixtral)
- **Versátil:** Adecuada para diversas tareas de aprendizaje automático y computación general.

3. T4 GPU:

- **Opción básica:** La T4 GPU ofrece un rendimiento decente con 8 TFLOPS en FP32 y 16 TFLOPS en FP16.
- **Suficiente para modelos pequeños:** Puede ser suficiente para fine-tuning de modelos de menor tamaño. (BERT)

4. CPU:

- **No recomendable:** La CPU no es adecuada para fine-tuning de LLMs grandes, ya que su rendimiento es significativamente inferior al de las opciones de GPU y TPU.

Resumen:

- **TPUv4:** La mejor opción para un alto rendimiento y eficiencia energética.
- **A100 GPU:** Buena alternativa con buen rendimiento y versatilidad.
- **T4 GPU:** Opción básica para modelos pequeños y presupuestos limitados.
- **CPU:** No se recomienda para fine-tuning de LLMs grandes.
- La disponibilidad de las opciones de GPU puede variar según la región y la hora del día.
- Los precios por hora de las GPUs pueden cambiar con el tiempo.

Conclusión:

La mejor opción para fine-tune_bert-base-uncased con ajuste de parámetros en Google Colab es la TPUv4. Sin embargo, si la disponibilidad de la TPUv4 es limitada o si el tiempo de fine-tuning es corto, la A100 GPU (**Mixtral**) o la T4 GPU (**BERT**) podrían ser alternativas viables, incluso la CPU.

Y se debe tener en cuenta:

- Evaluar el tiempo de fine-tuning para considerar el uso gratuito y los costos por hora.
- Consultar la disponibilidad de aceleradores en tiempo real en Google Colab.
- Monitorear el uso de recursos durante el proceso de fine-tuning.

Ejemplo de entorno de configuración

El primer paso es asegurarnos de que estamos usando una GPU T4 en Google Colab.

1. Abrir un nuevo notebook y seleccionar la pestaña de tiempo de ejecución en la parte superior.
2. Hacer clic en cambiar el tipo de tiempo de ejecución y seleccionar la opción T4 GPU.

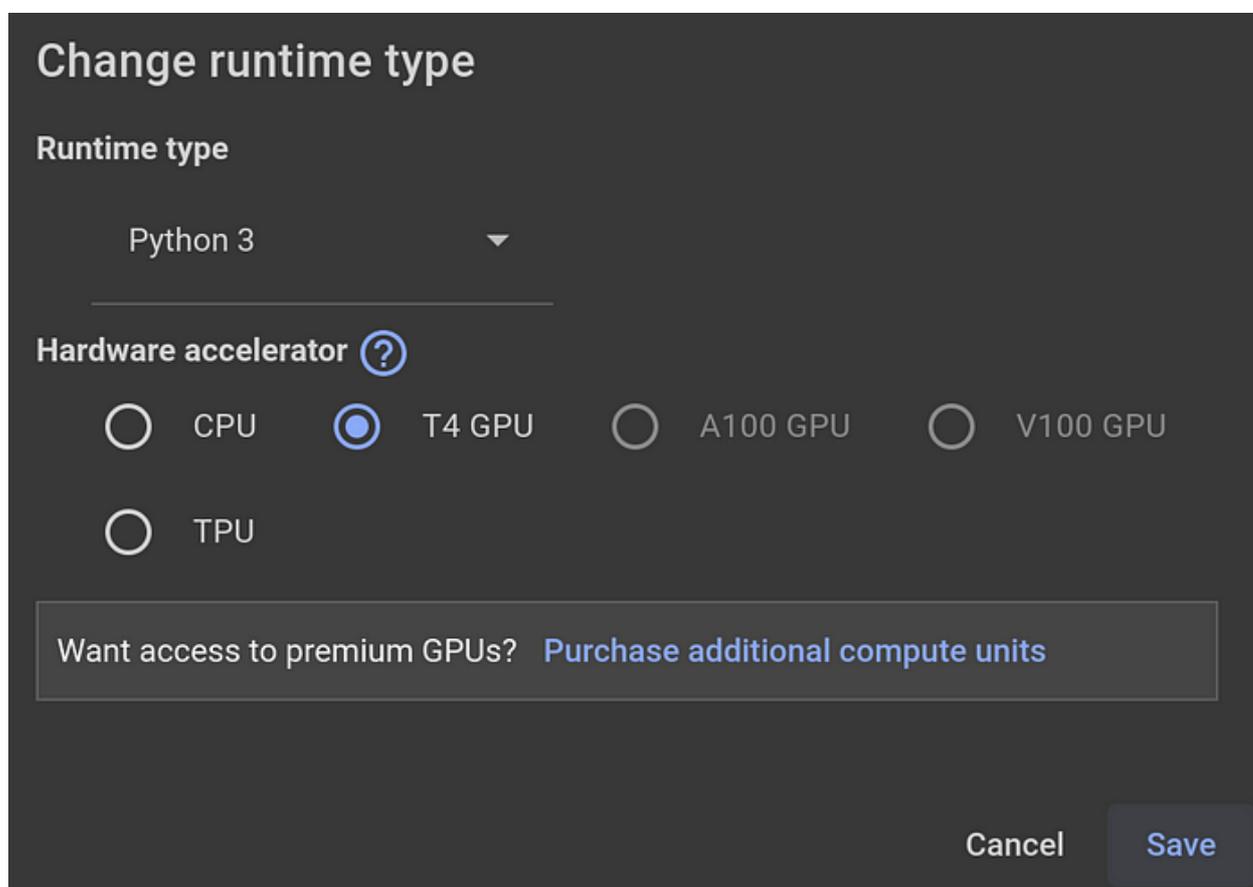


Imagen 30: Tipo de tiempo de ejecución en Google Colab.

3. A continuación, compruebe su instancia de GPU ejecutando:

!nvidia-smi

```
+-----+
| NVIDIA-SMI 535.104.05                Driver Version: 535.104.05   CUDA Version: 12.2   |
+-----+-----+-----+-----+-----+-----+
| GPU  Name          Persistence-M   Bus-Id        Disp.A   Volatile Uncorr. ECC  |
| Fan  Temp   Perf          Pwr:Usage/Cap     Memory-Usage   GPU-Util  Compute M.  |
|====+=====+====+=====+=====+=====+=====+
|  0   Tesla T4            Off          00000000:00:04:0  Off      0             |
| N/A   39C    P8             9W / 70W         0MiB / 15360MiB  0%          Default    |
|                                             MIG M.       N/A        |
+-----+-----+-----+-----+-----+-----+
| Processes:                               GPU Memory  |
|  GPU   GI    CI          PID    Type    Process name      Usage      |
|====+=====+====+=====+=====+=====+=====+
| No running processes found                |
+-----+-----+-----+-----+-----+-----+
+-----+
```

Imagen 31: Instancia de GPU, T4 GPU.

4. Ahora estamos listos. A continuación, instalamos las dependencias.

!pip install datasets transformers

Tenemos nuestro entorno configurado y listo para funcionar. Ahora, es el momento de preparar el conjunto de datos que vamos a utilizar para el entrenamiento (fine-tuning).

10.6. Anexo VI: Inferencia de LLMs [6] [21] [37]

Inferencia de un LLM

La inferencia de un LLM (*Large Language Model*) no se considera específicamente como fine-tuning en el sentido tradicional del término. Fine-tuning generalmente se refiere al proceso de ajustar un modelo preentrenado en una tarea específica mediante la exposición a datos de entrenamiento adicionales, generalmente con capas adicionales o un conjunto de datos específico para la tarea.

En cambio, la inferencia de un LLM generalmente implica utilizar el modelo preentrenado tal como está para generar secuencias de texto o realizar tareas de procesamiento del lenguaje natural (PLN) sin ajustar los pesos del modelo. En este caso, el modelo ya ha sido entrenado en una amplia gama de datos y tareas, y se utiliza para generar texto o realizar inferencia en nuevos datos sin modificar sus pesos.

Sin embargo, en algunos casos, la inferencia de un LLM puede considerarse una forma muy leve de fine-tuning si se están ajustando algunas partes del modelo para adaptarlo mejor a la tarea específica en cuestión, como ajustar los pesos de la última capa de salida para una tarea de generación de texto específica. Pero este ajuste generalmente es mínimo en comparación con el fine-tuning completo realizado en tareas específicas de aprendizaje automático.

Ajustar los parámetros como indicaciones contextuales (prompt), muestreo top-k, temperatura, longitud máxima, etc., cuando se utiliza un modelo de lenguaje grande (LLM) para la inferencia no se considera fine-tuning.

El fine-tuning se refiere al proceso de ajustar los pesos de un modelo preentrenado en una tarea específica utilizando datos específicos de esa tarea. Por ejemplo, si se toma un LLM preentrenado y se ajustan sus pesos en un conjunto de datos de clasificación de sentimientos, entonces se está realizando fine-tuning. Este proceso implica modificar los parámetros internos del modelo para adaptarlo mejor a una tarea particular.

Por otro lado, cuando se ajustan los parámetros como indicaciones contextuales, muestreo top-k, temperatura, longitud máxima, etc., se está modificando cómo se genera la salida del modelo durante la inferencia, pero no se están modificando los pesos internos del modelo en sí. En este caso, se está controlando cómo se utiliza el modelo preentrenado para generar texto o realizar tareas de procesamiento del lenguaje natural (PLN), pero no se está modificando el modelo en sí mismo mediante el ajuste de los pesos internos. Por lo tanto, no se considera fine-tuning.

Ajuste de un LLM con ajuste de prompt y parámetros de generación

Ajustar solo el prompt y los parámetros de generación para un LLM en una empresa de servicios específica (Canal de Isabel II) puede ser una forma simplificada de ajuste fino, pero tiene limitaciones:

Ventajas:

- **Simplicidad:** No se requiere recopilar ni preparar un conjunto de datos completo.
- **Rapidez:** El proceso de ajuste fino es más rápido que el ajuste fino tradicional.
- **Flexibilidad:** Se puede adaptar el modelo a diferentes tareas con relativa facilidad.

Desventajas:

- **Limitaciones de rendimiento:** El ajuste fino solo del prompt y los parámetros de generación puede no ser suficiente para lograr un rendimiento óptimo en todas las tareas.
- **Menor generalización:** El modelo puede ser menos adaptable a nuevos tipos de datos o tareas.
- **Dificultad para evaluar el rendimiento:** Es más difícil evaluar el rendimiento del modelo de forma precisa.

Enfoque basado en prompt y parámetros:

- **Ajuste del prompt:** Ajustar el prompt para guiar al modelo hacia la generación de texto deseado.
- **Ajuste de parámetros:** Ajustar parámetros para controlar la creatividad, la fluidez y la coherencia del texto generado.

En estos casos, los ajustes actúan como "**controles**" para influir en la salida del modelo sin alterar su estructura o pesos aprendidos. Se considera **inferencia con ajustes**.

Casos de uso:

- **Generación de respuestas a preguntas frecuentes (FAQ):** Ajustar el prompt y los parámetros para generar respuestas precisas y concisas a preguntas comunes.
- **Redacción de informes:** Ajustar el prompt y los parámetros para generar informes estructurados y concisos a partir de datos.
- **Creación de contenido creativo:** Ajustar el prompt y los parámetros para generar diferentes formatos de texto creativo, como poemas, guiones o código.

Recomendaciones:

- **Evaluar cuidadosamente el rendimiento:** Evaluar el rendimiento del modelo en diferentes escenarios y tareas para determinar su efectividad.
- **Combinar con otras técnicas:** Considerar combinar el ajuste del prompt y los parámetros con otras técnicas de ajuste fino, como el fine-tuning con un conjunto de datos específico, para obtener un mejor rendimiento.
- **Monitorear y ajustar:** Monitorizar el rendimiento del modelo en producción y realizar ajustes finos adicionales según sea necesario.

Conclusión:

Ajustar solo el prompt y los parámetros de generación puede ser una forma viable de ajustar un LLM para la empresa Canal de Isabel II, especialmente para tareas simples y bien definidas. Sin embargo, es importante considerar las limitaciones de este enfoque y evaluar cuidadosamente el rendimiento del modelo antes de su implementación en un entorno de producción. La recomendación es considerar técnicas de ajuste fino más completas para obtener un mejor rendimiento general.

10.7. Anexo VII: Fine-tuning a grandes modelos de lenguaje ^{[10] [11] [12] [13] [14] [15]}

Técnicas de fine-tuning

El fine-tuning de grandes modelos de lenguaje como BERT o Mixtral puede realizarse utilizando diversas técnicas para adaptar el modelo a tareas específicas o conjuntos de datos particulares. Algunas de estas técnicas incluyen:

1. Fine-tuning supervisado: Este enfoque implica ajustar los pesos del modelo preentrenado utilizando un conjunto de datos etiquetado para una tarea específica. Por ejemplo, en el caso de BERT, se puede realizar fine-tuning para tareas de clasificación de texto, extracción de información o generación de texto, entre otros.
2. Transfer learning: Esta técnica implica tomar un modelo preentrenado en una tarea general (como el modelado de lenguaje) y transferir el conocimiento aprendido a una tarea específica mediante fine-tuning. Esto es especialmente útil cuando se dispone de pocos datos etiquetados para la tarea específica, ya que el modelo preentrenado ya ha capturado una gran cantidad de conocimiento del lenguaje.
3. Ajuste de hiperparámetros: Durante el proceso de fine-tuning, se pueden ajustar los hiperparámetros del modelo, como la tasa de aprendizaje, el tamaño del lote, la función de pérdida, la arquitectura del modelo y otros, para mejorar el rendimiento en la tarea objetivo.
4. Data augmentation: Esta técnica implica aumentar la cantidad de datos de entrenamiento mediante la aplicación de transformaciones aleatorias o semánticamente relevantes a los ejemplos existentes. Esto ayuda a mejorar la generalización del modelo y a reducir el riesgo de sobreajuste.
5. Ensemble methods: Se pueden combinar múltiples modelos finetuned para formar un ensamble, lo que puede mejorar el rendimiento general del sistema al aprovechar la diversidad de los modelos individuales.
6. Regularización: Se pueden aplicar técnicas de regularización, como la dropout o la penalización de los pesos, para evitar el sobreajuste y mejorar la capacidad de generalización del modelo.

7. **Afinación por capas:** En lugar de ajustar todos los parámetros del modelo, se pueden congelar algunas capas y solo ajustar los parámetros de las capas superiores durante el fine-tuning. También se pueden agregar capas adicionales al final del clasificador de salida del modelo. Esto puede ser útil cuando se dispone de pocos datos de entrenamiento o se quiere evitar el sobreajuste.

Estas son algunas de las técnicas comunes que se pueden aplicar para hacer fine-tuning a grandes modelos de lenguaje como BERT o Mixtral, con el objetivo de adaptarlos a tareas específicas y mejorar su rendimiento en dominios de aplicación particulares.

Cómo realizar un ajuste fino de un LLM para un conjunto de datos específico de una empresa con preguntas y respuestas:

Pasos generales:

1. Recopilación y preparación de datos
 - **Recopilar:** Obtener un conjunto de datos de preguntas y respuestas de la empresa. Este conjunto de datos debe ser lo más representativo posible de las consultas que el modelo deberá responder en el futuro.
 - **Preprocesamiento:** Limpiar y preprocesar los datos para eliminar ruido, corregir errores y normalizar el formato.
 - **Dividir:** Dividir el conjunto de datos en conjuntos de entrenamiento, validación y/o prueba.
2. Elección del modelo LLM
 - **Seleccionar:** Elegir un modelo LLM pre-entrenado adecuado para la tarea de preguntas y respuestas. Algunos modelos populares incluyen BERT, RoBERTa, XLNet o T5 (*Text-To-Text Transfer Transformer*).
 - **Descargar:** Descargar el modelo y el tokenizador. El modelo contiene los parámetros, los pesos que han sido ajustados durante el proceso de entrenamiento y la arquitectura de la red neuronal, mientras que el tokenizador es responsable de dividir el texto en unidades más pequeñas, como palabras o subpalabras, para que el modelo pueda entenderlo.

3. Ajuste fino del modelo

- **Configurar:** Configurar el proceso de ajuste fino, incluyendo la función de pérdida, el optimizador, la tasa de aprendizaje y el número de épocas.
- **Entrenar:** Entrenar el modelo LLM en el conjunto de entrenamiento, utilizando el conjunto de validación para monitorizar el rendimiento y evitar el sobreentrenamiento.
- **Evaluar:** Evaluar el rendimiento del modelo ajustado en el conjunto de validación y/o prueba.

4. Implementación

- **Integrar:** Integrar el modelo ajustado en la aplicación o servicio de la empresa.
- **Monitorizar:** Monitorizar el rendimiento del modelo en producción y realizar ajustes finos adicionales si es necesario.

Consideraciones:

- El ajuste fino de un LLM puede ser un proceso complejo y computacionalmente costoso.
- Es importante seleccionar el modelo LLM adecuado y configurar correctamente el proceso de ajuste fino para obtener un buen rendimiento.
- Es importante evaluar cuidadosamente el rendimiento del modelo ajustado antes de su implementación en un entorno de producción.

Conclusión:

El ajuste fino de un LLM para un conjunto de datos específico de una empresa con preguntas y respuestas puede ser una herramienta poderosa para mejorar la precisión y la eficiencia de las tareas de procesamiento del lenguaje natural. Con una cuidadosa planificación y ejecución, se puede obtener un modelo que satisfaga las necesidades específicas de la empresa Canal de Isabel II.

10.8. Anexo VIII: PEFT (Ajuste fino eficiente en parámetros) ^[17]^[18]

Introducción a PEFT

- PEFT (*Prefix Engineering for Fine-Tuning*), o Ajuste Fino Eficiente en Parámetros, es una técnica diseñada para ajustar de manera eficiente modelos de lenguaje preentrenados a aplicaciones específicas.
- Su objetivo es reducir los costos computacionales y de almacenamiento asociados con el ajuste completo de modelos, al tiempo que mantiene un rendimiento comparable.

Problemas con el Ajuste Completo

- Grandes pesos de entrenamiento.
- Requerimientos de potencia de cálculo.
- Aumento en el tamaño del archivo de modelos.

Métodos PEFT

- **LoRA**: Ajusta un pequeño número de parámetros adicionales del modelo. Lora se enfoca en modificar solo los prefijos (o tokens iniciales) del modelo.
- **Afinación P** (*Quick Layer-wise Fine-Tuning*): Ajuste rápido de parámetros de una o más capas específicas del modelo.
- **Ajuste Rápido** (*Fast Fine-Tuning*): Métodos de afinación rápida. Puede incluir estrategias como ajustar solo las últimas capas del modelo o utilizar tasas de aprendizaje más altas para converger más rápidamente.
- **QLoRA** (*Quick Layer-wise Relevance Adjustment*): Afinación de parámetros específicos de capas para adaptar el modelo a una tarea específica.

Características Principales

- Integrado con bibliotecas como Transformers, Diffusers y Accelerate.
- Proporciona una forma más rápida y sencilla de cargar, entrenar y utilizar modelos grandes para la inferencia.
- Reduce los costos computacionales y de almacenamiento al ajustar solo un pequeño número de parámetros.

Cómo Funciona PEFT

- Define una clase **PeftConfig** para cada método PEFT, que almacena los parámetros importantes para crear un **PeftModel**.
- Especifica los parámetros relevantes, como el tipo de tarea, el modo de inferencia, la dimensión de las matrices de rango bajo y la probabilidad de abandono.

Ventajas de PEFT

- Permite adaptar eficientemente grandes modelos preentrenados a diferentes aplicaciones posteriores.
- Facilita el entrenamiento y el uso de modelos grandes en hardware de consumo.
- Ofrece un rendimiento comparable al ajuste completo de modelos, pero con una fracción de los recursos.

LoRA (Ajuste de Rango Bajo)

- LoRA es un método utilizado en PEFT que ayuda a reducir el número de parámetros entrenables en un modelo preentrenado.
- En lugar de ajustar todos los parámetros del modelo, LoRA ajusta solo un subconjunto de parámetros de rango bajo, lo que reduce significativamente los costos de entrenamiento y almacenamiento.

QLoRA (Ajuste de Rango Bajo Cuantizado)

- QLoRA es una variante de LoRA que utiliza técnicas de cuantización para comprimir aún más los parámetros de rango bajo.
- Al aplicar la cuantización, QLoRA reduce la precisión de los parámetros de rango bajo, lo que permite un mayor grado de compresión sin comprometer demasiado el rendimiento del modelo.
- El secreto detrás de QLoRA radica en la cuantización de 4 bits. En lugar de utilizar representaciones de alta precisión, los parámetros del modelo se comprimen a solo 4 bits.
- La cuantización reduce la cantidad de memoria necesaria para almacenar los pesos del modelo, lo que es crucial para su eficiencia.

Relación entre PEFT, LoRA y QLoRA

- PEFT incorpora métodos como LoRA y QLoRA como parte de su enfoque para el ajuste eficiente de modelos preentrenados.
- Estos métodos permiten reducir drásticamente el número de parámetros entrenables en el modelo, lo que a su vez reduce los costos computacionales y de almacenamiento asociados con el entrenamiento y el despliegue del modelo.

Conclusión

- PEFT es una técnica para reducir los costos asociados con el ajuste de modelos de lenguaje preentrenados. Utiliza métodos como LoRA y QLoRA para lograr un ajuste fino eficiente en parámetros, lo que hace que los modelos preentrenados sean más accesibles y prácticos para una amplia gama de aplicaciones.

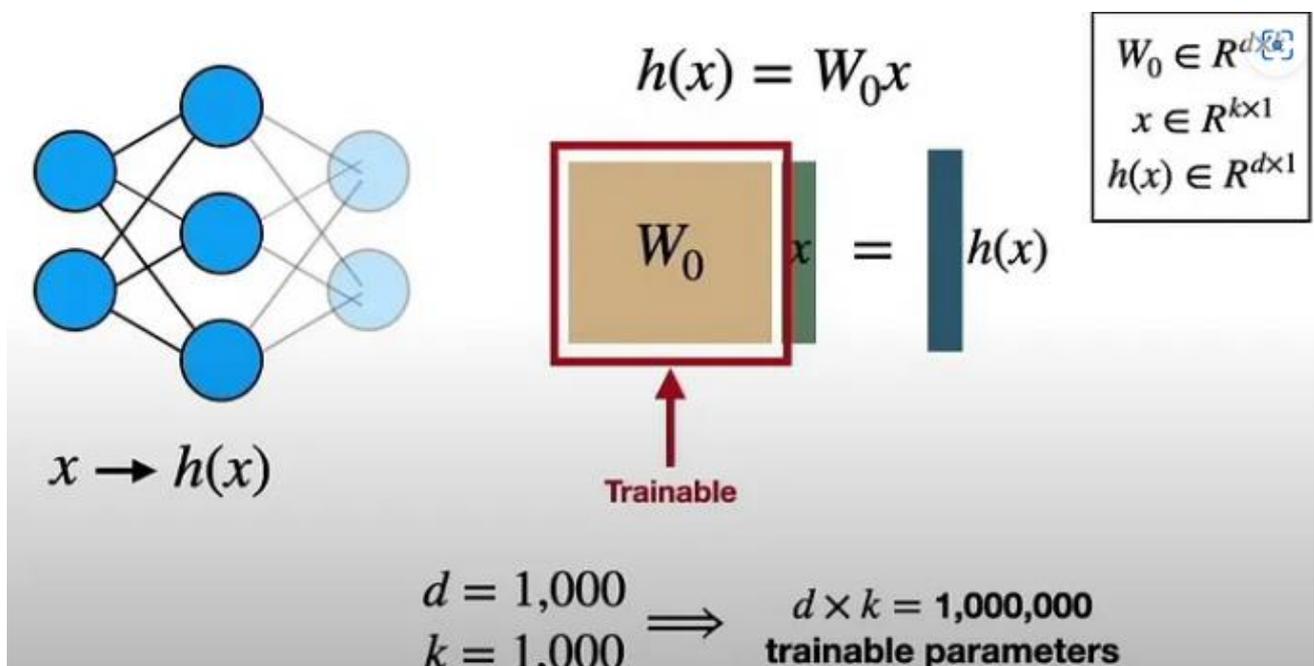


Imagen 32: Ajuste fino completo de los modelos sin enfoque PEFT. <https://medium.com/@levxn/lora-and-qlora-effective-methods-to-fine-tune-your-llms-in-detail-6e56a2a13f3c>

10.9. Anexo IX: LoRA y QLoRA: Cuantización de modelos de lenguaje ^{[16] [18] [19] [20] [38]}

LoRA (Low Rank Adapters): Adaptadores de bajo rango para fine-tuning.

LoRA es un método de fine-tuning que utiliza matrices de bajo rango para ajustar los pesos de un modelo pre-entrenado. En lugar de ajustar directamente todos los millones de parámetros del modelo, LoRA se centra en modificar una pequeña cantidad de vectores de bajo rango, lo que reduce significativamente la cantidad de parámetros a ajustar.

El fine-tuning es una técnica crucial para adaptar modelos de lenguaje pre-entrenados (LLMs) a tareas específicas. Sin embargo, el fine-tuning tradicional puede ser computacionalmente costoso y requerir grandes cantidades de datos. El método "*Low Rank Adapters*" (LoRA) surge como una alternativa eficiente para el fine-tuning de LLMs.

Fine-tuning de LoRA:

1. **Descomposición de Matriz:** Se descompone la matriz de pesos del modelo pre-entrenado en una matriz base y una matriz de adaptación de bajo rango.
2. **Adaptación de bajo rango:** Se ajusta la matriz de adaptación de bajo rango para la tarea específica.
3. **Integración:** Se multiplica la matriz base por la matriz de adaptación de bajo rango ajustada para obtener los nuevos pesos del modelo.

LoRA se ha convertido en una herramienta valiosa para el fine-tuning de LLMs, ofreciendo una alternativa eficiente y escalable al fine-tuning tradicional. Su capacidad para reducir la cantidad de parámetros a ajustar lo convierte en una opción atractiva para tareas que requieren un alto rendimiento con recursos computacionales limitados.

Sin embargo, es importante considerar las ventajas y desventajas de LoRA al elegir un método de fine-tuning para un caso de uso específico. La elección del método adecuado dependerá de la tarea, el tamaño del modelo, los recursos disponibles y los objetivos de rendimiento.

QLoRA: Cuantización (*Quantized*) de LoRA.

QLoRA (*Quantized LoRA*) es una extensión de LoRA que combina el fine-tuning con la cuantización. La cuantización reduce la precisión de los pesos del modelo para mejorar la eficiencia de la inferencia, es decir, la velocidad con la que el modelo puede generar predicciones.

Cuantización con QLoRA:

1. Cuantización de la matriz base: Se cuantiza la matriz base del modelo pre-entrenado a un formato de menor precisión.
2. Fine-tuning con LoRA: Se realiza el fine-tuning de la matriz de adaptación de bajo rango utilizando LoRA.
3. Fusión: Se fusionan la matriz base cuantizada y la matriz de adaptación de bajo rango ajustada para obtener el modelo cuantizado.

LoRA y QLoRA son **métodos complementarios** para mejorar la eficiencia de los LLMs. LoRA se centra en la **reducción del número de parámetros a ajustar durante el fine-tuning**, mientras que QLoRA se centra en la **reducción de la precisión de los pesos del modelo para mejorar la velocidad de inferencia**.

Se puede combinar LoRA y QLoRA para obtener un modelo LLM eficiente tanto en términos de fine-tuning como de inferencia. Esto permite ajustar el modelo a una tarea específica con un número reducido de parámetros y una precisión de inferencia optimizada.

La elección de LoRA, QLoRA o una combinación de ambos dependerá de las necesidades específicas del proyecto, como la precisión requerida, la velocidad de inferencia y los recursos disponibles.

Ventajas y desventajas de LoRA y QLoRA

- **Ventajas de LoRA:**
 1. Eficiencia computacional: Requiere menos parámetros a ajustar, reduciendo el tiempo y los recursos computacionales necesarios.
 2. Generalización: Puede adaptarse a una variedad de tareas con relativamente pocos datos.
 3. Escalabilidad: Funciona bien con modelos de lenguaje de gran tamaño.
- **Desventajas de LoRA:**
 1. Rendimiento: En algunos casos, el rendimiento del fine-tuning con LoRA puede ser ligeramente inferior al del fine-tuning tradicional.
 2. Complejidad: La implementación de LoRA puede ser más compleja que el fine-tuning tradicional.
- **Ventajas de QLoRA:**
 1. Eficiencia de inferencia: Reduce el tamaño del modelo y la cantidad de operaciones de cálculo necesarias para la inferencia.
 2. Reducción de memoria: Disminuye la cantidad de memoria requerida para almacenar el modelo.
- **Desventajas de QLoRA:**
 1. Rendimiento: Puede afectar ligeramente la precisión del modelo durante la inferencia.
 2. Complejidad: La implementación de QLoRA puede ser más compleja que la de LoRA.

Funcionamiento de Lora y QLoRa

LoRa agrega una pequeña cantidad de parámetros entrenables, es decir, adaptadores, para cada capa del LLM y congela todos los parámetros originales. Para el ajuste fino, solo tenemos que actualizar los pesos del adaptador, lo que reduce significativamente la huella de memoria.

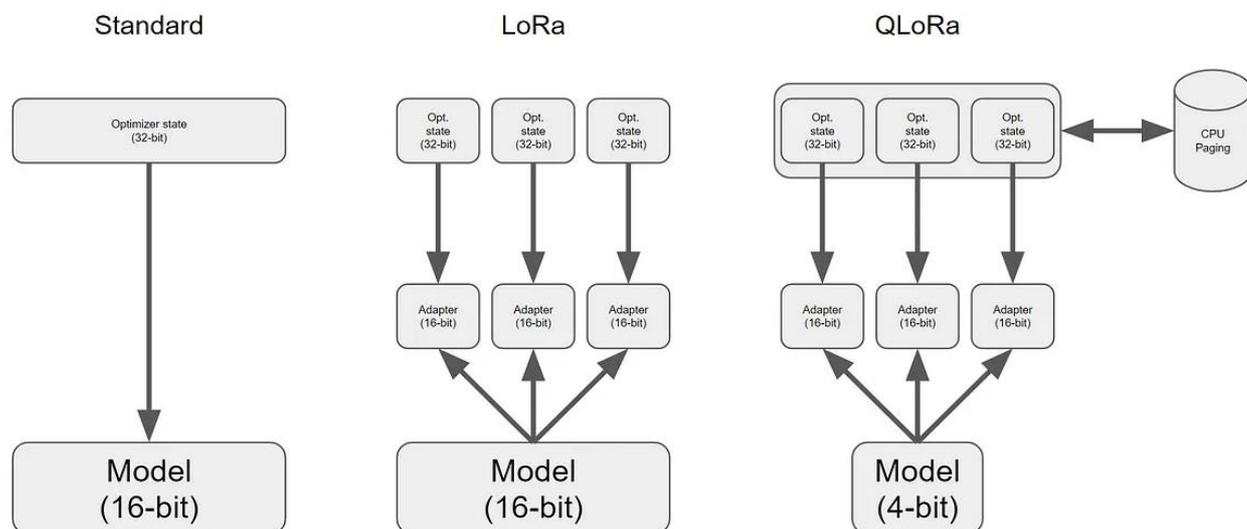


Imagen 33: Ajuste de modelo de lenguaje grande en la GPU.

<https://kaitchup.substack.com/p/qlora-fine-tune-a-large-language-model-on-your-gpu-27bed5a03e2b>

QLoRa lleva su enfoque un paso más allá al introducir tres técnicas clave: cuantización de 4 bits, cuantización doble y la explotación de la memoria unificada NVIDIA para la paginación.

En pocas palabras, cada uno de estos pasos funciona de la siguiente manera:

- **Cuantización NormalFloat de 4 bits:** Este es un método que mejora la cuantización de cuantiles. Garantiza un número igual de valores en cada intervalo de cuantificación. De este modo, se evitan problemas de cálculo y errores para los valores atípicos.
- **Doble cuantización:** Utiliza dos niveles de cuantización: uno para los pesos y otro para las activaciones. Esto permite una mayor compresión y eficiencia en el modelo.
- **Paginación con memoria unificada:** Se basa en la función de memoria unificada de NVIDIA y gestiona automáticamente las transferencias de página a página entre la CPU y la GPU. Garantiza un procesamiento de GPU sin errores, especialmente en situaciones en las que la GPU puede quedarse sin memoria.

Todos estos pasos reducen drásticamente los requisitos de memoria para el ajuste fino, al tiempo que funcionan casi a la par con el ajuste fino estándar.

10.10. Anexo X: Respuestas obtenidas

BERT_fine-tuning: Preguntas y Respuestas del Canal de Isabel II

Autor: [Antonio Fernández](#) de [Canal de Isabel II](#). Formación: [Grado Ingeniería Informática](#) Aplicación desarrollada para TFG_Inteligencia_Artificial. Proporcione una pregunta y el asistente encontrará la respuesta.

<p>Question</p> <p>¿Cuál es el teléfono de atención al cliente?</p> <p>Clear Submit</p>	<p>Answer</p> <p>900 365 365</p>
--	---

BERT_fine-tuning: Preguntas y Respuestas del Canal de Isabel II

Autor: [Antonio Fernández](#) de [Canal de Isabel II](#). Formación: [Grado Ingeniería Informática](#) Aplicación desarrollada para TFG_Inteligencia_Artificial. Proporcione una pregunta y el asistente encontrará la respuesta.

<p>Question</p> <p>¿Qué puedo avisar al servicio de incidencias?</p> <p>Clear Submit</p>	<p>Answer</p> <p>cortes de agua o fugas</p>
---	--

BERT_fine-tuning: Preguntas y Respuestas del Canal de Isabel II

Autor: [Antonio Fernández](#) de [Canal de Isabel II](#). Formación: [Grado Ingeniería Informática](#) Aplicación desarrollada para TFG_Inteligencia_Artificial. Proporcione una pregunta y el asistente encontrará la respuesta.

<p>Question</p> <p>¿A qué número puedo llamar para reportar una incidencia en el suministro de gas?</p> <p>Clear Submit</p>	<p>Answer</p> <p>Lo siento, no estoy seguro de la respuesta. Por favor, reformula tu pregunta sobre Averías o Contacto del Canal de Isabel II.</p>
--	---

Imagen 34: BERT_fine-tuning. Preguntas y Respuestas del Canal de Isabel II.

Mixtral 8B. TFG: Asistente de usuarios/clientes de Canal de Isabel II

Autor: [Antonio Fernández](#) de [Canal de Isabel II](#). Formación: [Grado Ingeniería Informática](#). Aplicación desarrollada para TFG



Tengo poca presión en mi vivienda, ¿qué puedo hacer?

Lo siento por la situación que está experimentando con la presión de agua en su vivienda. A continuación, le ofrezco algunas sugerencias que puede considerar:

1. Verifique si la baja presión de agua es un problema general en su vecindario o únicamente en su hogar. Puede preguntar a sus vecinos si también están experimentando el mismo problema. Si es un problema general, puede comunicarse con Canal de Isabel II al número de teléfono 900 365 365 para informar sobre el problema y solicitar asistencia.
2. Si el problema es únicamente en su hogar, es posible que la presión del agua se vea afectada por la altura de su vivienda. Si vive en un edificio de varias plantas, la presión puede disminuir en los pisos superiores. En este caso, puede considerar la posibilidad de instalar una bomba de presión de agua para aumentar la presión del agua en su hogar.
3. Otra causa posible de baja presión de agua puede ser una fuga en la tubería de suministro de agua. Si sospecha que hay una fuga, puede verificar si hay humedad o manchas en las paredes, suelos o techos, o si hay un ruido de flujo de agua constante en las tuberías. Si encuentra una fuga, debe comunicarse con un fontanero profesional para que la repare.
4. También puede verificar si hay algún grifo o grifo que gotee en su hogar, ya que esto puede disminuir la presión del agua en el resto de la vivienda. Si encuentra algún grifo que gotee, asegúrese de cerrar completamente la llave de paso y, si es necesario, reemplace las piezas.



¿Qué parámetros definen la calidad del agua?

Enviar

Repetir

Deshacer

Borrar

Imagen 35: Asistente de usuarios/clientes de Canal de Isabel II. <https://antonio49-chat-mixtral-aguausuarioconhilos.hf.space>

Mixtral 8B. TFG: Experto en sector del agua del Canal de Isabel II

Autor: [Antonio Fernández](#) de [Canal de Isabel II](#). Formación: [Grado Ingeniería Informática](#). Aplicación desarrollada para TFG



Que diametros son los apropiados para una tubería de fundición dúctil

Hola! Como asistente de escritorio, estoy aquí para brindarte información sobre el Canal de Isabel II y los servicios de abastecimiento, depuración, reutilización, alcantarillado y calidad del agua.

En cuanto a las tuberías de fundición dúctil, estas son una opción popular para el transporte de agua potable y aguas residuales debido a su durabilidad y resistencia a la corrosión. Los diámetros adecuados para una tubería de fundición dúctil dependen de varios factores, como la presión hidrostática, el caudal y las características del terreno donde se instalará la tubería.

En general, los diámetros más comunes para las tuberías de fundición dúctil en el transporte de agua potable oscilan entre 50 y 600 mm, aunque también se pueden encontrar tuberías de mayor diámetro en algunas aplicaciones especiales. Para el transporte de aguas residuales, los diámetros más comunes suelen ser mayores, oscilando entre 150 y 1200 mm.

Es importante tener en cuenta que los diámetros de las tuberías deben ser seleccionados cuidadosamente para garantizar una distribución adecuada del flujo de agua y minimizar las pérdidas de presión. Además, las tuberías de fundición dúctil deben instalarse y mantenerse correctamente para garantizar una vida útil prolongada y minimizar los riesgos de fugas y roturas.

En cuanto a las obras hidráulicas y el cumplimiento de la sanidad y la seguridad de las instalaciones



Imagen 36: Experto en sector del agua del Canal de Isabel II. <https://antonio49-chat-mixtral-expertoconhilos.hf.space>

10.11. Anexo XI: Comparativa con LLM [22] [23]

- <https://huggingface.co/chat>
- <https://huggingface.co/chat/assistants>
- <https://huggingface.co/chat/assistants?user=Antonio49>
- [https://hf.co/chat/assistant/6617a6777cab34f10b277e00:](https://hf.co/chat/assistant/6617a6777cab34f10b277e00)

 Google-gemma-Asistente usuarios de la empresa Canal de Isabel II

¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?



Sí. Si un usuario o cliente de Channel of Spain Second does not pay a an invoice for water services provided by the company will cut off their supply as part **their payment recovery policy**.

¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?



Para restaurar el servicios después del cortes debido al non-pago:

- Contacte con Customer Service en 901 or through online chat in Office Virtual to make arrangements and schedule reconnection on paying outstanding balance including late fees if applicable

Imagen 37: Inferencia al modelo LLM google/gemma-1.1-7b-it desde HuggingChat.

- <https://hf.co/chat/assistant/66179d6942adca7f41a4f4f1>:

 Mistral-7B-Instruct-v0.2-Asistente usuarios de la empresa Canal de Isabel II

¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?



Sí, en ciertas circunstancias, Canal de Isabel II puede cortar el suministro de agua a un usuario o cliente que no pague sus facturas. Esto se realiza en virtud de las normas legales y reguladas por el gobierno español. Sin embargo, antes de hacerlo, el proveedor solicitará repetidamente que se paguen las deudas y se tomarán otras medidas, como la aplicación de multas o el trámite de congelamiento de cuentas bancarias. En última instancia, solo como medida extrema y cuando se ha agotado todo otro medio, el corte del suministro se considerará.

Si usted está experimentando dificultades para pagar una factura actual o antecedentes financieros, es recomendable contactarlo inmediatamente con Canal de Isabel II para abordar la situación y buscar una solución amigable con la compañía. Pueden ofrecerle planificar un pago gradual o una amplia gama de opciones de asistencia social si es necesario.

El número de teléfono de atención al cliente de Canal de Isabel II es 902 111 311. También puede acceder a su cuenta en línea en <https://www.canaldeisabelsegunda.es/clientes/portal-cliente/inicio> para ver sus facturas y hacer pagos online.

¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?



Si ha tenido un corte de suministro de agua debido a una factura no pagada, siga estos pasos para restablecer el servicio:

1. Pague la factura deuda. Puede hacerlo a través de varios medios, como:

- Pagando en persona en una sucursal de Canal de Isabel II.
- Haciendo un pago en línea en su cuenta de cliente en <https://www.canaldeisabelsegunda.es/clientes/portal-cliente/inicio>.
- Realizando un pago telefónico llamando al número de atención al cliente de Canal de Isabel II al 902 111 311.

Imagen 38: Inferencia al modelo LLM mistralai/mistral-7B-Instruct-v0.2 desde HuggingChat.

- <https://hf.co/chat/assistant/6617a14ae0641b76ef1fb97e>:

 Mistral-8Bx7B-Instruct-v0.1-Asistente usuarios de la empresa Canal de Isabel II

¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?



Sí, es posible que Canal de Isabel II corte el suministro de agua en caso de impago de facturas. Según su normativa, tras el vencimiento del plazo de pago y sin que se haya satisfecho la deuda, la empresa enviará un requerimiento de pago al cliente. Si transcurridos quince días hábiles desde la recepción de dicho requerimiento persiste el impago, Canal de Isabel II podrá interrumpir el suministro. No obstante, antes de llegar a esta situación extrema, la compañía ofrece facilidades de pago y otros mecanismos para ayudar a sus clientes a saldar sus deudas. Si tienes alguna duda o necesitas asistencia con tu situación específica, te recomiendo que contactes directamente con Canal de Isabel II a través de su Oficina Virtual o por teléfono.

¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?



Si han cortado tu suministro de agua por impago de una factura, sigue estos pasos para restaurarlo:

1. Paga la deuda: Asegúrate de saldar la totalidad de la deuda pendiente, incluyendo cualquier recargo o gasto adicional relacionado con el corte y posterior restablecimiento del servicio. Generalmente, debes realizar este pago en la oficina más cercana de Canal de Isabel II o a través de su Oficina Virtual.
2. Solicita la revisión de la cuenta (si lo consideras necesario): Si crees que hubo un error en el cobro o que la situación merece ser reconsiderada, puedes solicitar la revisión de tu caso en las oficinas de Canal de Isabel II o a través de su servicio de atención al cliente.

Imagen 39: Inferencia al modelo LLM mistralai/Mixtral-8x7B-Instruct-v0.1 desde HuggingChat.

- <https://hf.co/chat/assistant/6617a81c440143a0437acccd>:



CohereForAI-Asistente usuarios de la empresa Canal de Isabel II

¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?



Sí, según las Condiciones Generales de Suministro de Agua aprobadas por la Comunidad de Madrid, Canal de Isabel II puede suspender o limitar el suministro de agua en caso de impago de las facturas dentro del plazo establecido.

Antes de proceder a cualquier suspensión o limitación del servicio, Canal de Isabel II realizará las notificaciones pertinentes al usuario mediante comunicaciones escritas informando sobre el impago y proporcionando un período de tiempo adicional para realizar el pago antes de llevar a cabo la interrupción del suministro. El usuario tendrá derecho a reclamar y proporcionar pruebas de que ha realizado el pago o que existen errores en la facturación.

Es importante destacar que la suspensión o limitación del suministro de agua es considerado como un último recurso después de haber agotado otros medios de cobro y notificación. La restauración del servicio luego de su suspensión podría incurrir en cargos adicionales que serán responsabilidad del usuario.

Si está experimentando dificultades para pagar sus facturas de agua, le recomendamos que se comunique con Canal de Isabel II lo antes posible para discutir posibles arreglos de pago alternativos u opciones disponibles para evitar la interrupción del servicio. Mantener una comunicación abierta y buscar soluciones tempranas generalmente resulta en mejores resultados tanto para el cliente como para la compañía de servicios públicos.

¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?



Para restablecer el servicio de agua después de un corte debido al impago de una factura, debe seguir estos pasos:

1. Pagar la factura pendiente: Lo primero que debe hacer es abonar el importe total adeudado correspondiente a la factura impaga. Puede realizar el pago utilizando los métodos aceptados por Canal de Isabel II, tales como transferencia bancaria, tarjeta de crédito o débito, domiciliación bancaria, entre

Imagen 40: Inferencia al modelo LLM CohereForAI/c4ai-command-r-plus desde HuggingChat.

- <https://hf.co/chat/assistant/6617a9f062f9e3e079076e64>:

 NousResearch/Mistral-8Bx7B-DPO-Asistente usuarios de la empresa Canal de Isabel II

¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?



Canal de Isabel II es una empresa que proporciona servicios de abastecimiento de agua en Madrid, España. De acuerdo con sus condiciones generales del contrato, sí pueden cortar el suministro de agua si un cliente mantiene impagos por dos meses consecutivos o tres alternos. Es importante que las personas cumplan con sus obligaciones de pago a tiempo para evitar posibles suspensiones de servicio. Si tienes dificultades financieras y crees que podrías no poder pagar una factura, te recomiendo contactar con Atención al Cliente de Canal de Isabel II para buscar soluciones antes de llegar a esta situación.

¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?



Si tu servicio ha sido suspendido debido a impagos de facturas de agua, lo primero que debes hacer es saldar la deuda correspondiente. Para esto, puedes llamar a Atención al Cliente de Canal de Isabel II (901 306 307), ingresar a su oficina virtual (<https://oficinavirtual.canaldeisabelsegunda.es/>) y realizar el pago mediante transferencia bancaria u otro medio de pago disponible. Una vez que hayas liquidado la deuda, comunícalo a Atención al Cliente indicando que deseas que se restablezca el servicio. Ten en cuenta que puede haber un período adicional para el procesamiento del pago y la inspección antes de que el servicio sea restaurado. Si necesitas información más detallada sobre cómo proceder, también puedes visitar el apartado "Suspensión temporal" dentro de las Condiciones Generales de Contratación en la página web oficial de Canal de Isabel II: <http://www.canaldeisabelsegunda.es/CONDICIONES-GENERALES.aspx>.

Imagen 41: Inferencia al modelo LLM NousResearch/Mixtral-8x7B-DPO desde HuggingChat.

- <https://huggingface.co/spaces/Antonio49/Chat.Mixtral.AguaUsuarioConHilos>:

Mixtral 8B. TFG: Asistente de usuarios/clientes de Canal de Isabel II

Autor: [Antonio Fernández](#) de [Canal de Isabel II](#). Formación: [Grado Ingeniería Informática](#). Aplicación desarrollada para TFG



¿Puede Canal de Isabel II cortar el suministro de agua si no pago una factura?

Sí, Canal de Isabel II, como empresa responsable del suministro de agua, tiene el derecho de interrumpir el servicio en caso de falta de pago por parte del usuario. Sin embargo, antes de tomar tal medida, la compañía está obligada a seguir un procedimiento establecido por ley, que incluye el envío de avisos y la realización de gestiones para buscar una solución amistosa al impago.



Es importante que, en caso de dificultades para afrontar el pago de una factura de agua, el usuario se ponga en contacto con Canal de Isabel II para informar de su situación y tratar de encontrar una solución adecuada. La compañía ofrece diferentes posibilidades de fraccionamiento de pago y otras ayudas a aquellos usuarios que se encuentren en situación de vulnerabilidad económica.

En cualquier caso, es recomendable mantener siempre al corriente el pago de las facturas de suministros básicos como el agua, ya que su interrupción puede generar graves problemas en el desarrollo de la vida diaria.



¿Cómo puedo restablecer el servicio, si he tenido un corte de suministro por no pagar una factura de agua?



Si tu suministro de agua ha sido interrumpido por impago de una factura de Canal de Isabel II, puedes restablecer el servicio siguiendo estos pasos:

1. Paga la deuda pendiente: El primer paso para restablecer el suministro es saldar la deuda que ha originado el corte. Puedes hacerlo a través de los diferentes canales de pago que ofrece la compañía, como la oficina virtual, el teléfono de atención al cliente, los cajeros automáticos o las oficinas comerciales.
2. Solicita el restablecimiento del servicio: Una vez abonada la deuda, debes contactar con Canal de Isabel II para solicitar la reactivación del suministro. Puedes hacerlo a través de la oficina virtual, llamando al teléfono de atención al cliente o acudiendo a una de las oficinas comerciales de la compañía.
3. Verifica la fecha y hora del restablecimiento del servicio: Una vez solicitado el restablecimiento del servicio, Canal de Isabel II te informará de la fecha y hora en la que se producirá la reactivación del suministro. Asegúrate de estar presente en el momento en que se realice el restablecimiento para comprobar que todo funciona correctamente.

Recuerda que, si te encuentras en situación de vulnerabilidad económica, puedes solicitar a Canal de Isabel II la aplicación de medidas especiales de aplazamiento o fraccionamiento de pago. Además, si has sufrido un corte de suministro por impago, es recomendable que establezcas un plan de pagos regular para evitar nuevos cortes en el futuro.

Imagen 42: Inferencia al modelo LLM Mixtral-8x7B-Instruct-v0.1 en el espacio creado.

10.12. Anexo XII: Diagrama de Despliegue

El Nodo principal es el “Servidor de Hugging Face”, que representa la infraestructura de Hugging Face donde se aloja el modelo fine-tuning de BERT Q&A Canal Isabel II y el modelo Mixtral-8x7B-Instruct-v0.1.

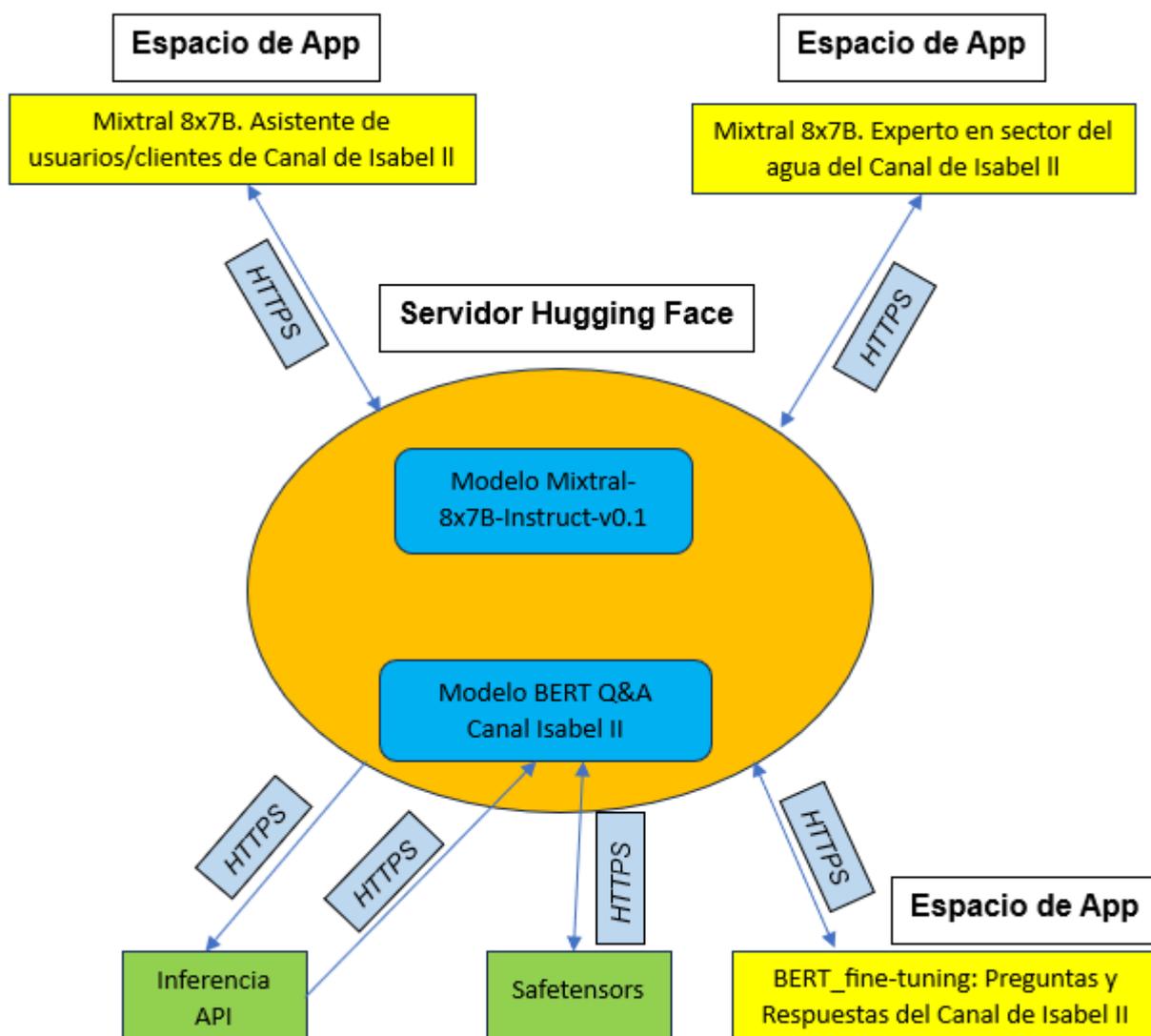


Imagen 43: Diagrama de despliegue en Hugging Face.

Hugging Face utiliza HTTPS (*Hypertext Transfer Protocol Secure*) para la comunicación interna con su API de inferencia cuando se utiliza Safetensors. Esto garantiza que la información se transmita de manera segura y protege contra posibles ataques o interceptaciones.


```

BertForQuestionAnswering(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(31002, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
)
(qa_outputs): Linear(
  in_features=768, out_features=2, bias=True
  (additional_layers): Sequential(
    (0): Linear(in_features=768, out_features=768, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.1, inplace=False)
    (3): Linear(in_features=768, out_features=768, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.1, inplace=False)
  )
)

```

Imagen 45: Estructura del modelo BERT con capas agregadas.

10.14. Anexo XIV: Código del proyecto

- Conjunto de datos en formato JSON que contiene preguntas y respuestas del Canal de Isabel II.
 - https://drive.google.com/file/d/1wvPeB7L8waVXeEzWHjAGEWdClvW3bgFB/view?usp=drive_link
- Notebook del proyecto para el Fine-Tuning del modelo BERT pre-entrenado. (Optimizador AdamW).
 - https://colab.research.google.com/drive/1e2-KS1DW1Xtmg4IU_nPJnl_h2NeARj_o

Índice de Notebook:

- [Trabajar con conjuntos de datos de preguntas y respuestas etiquetadas y modelos de Hugging Face](#)
- [Requisitos de hardware](#)
- [RAM](#)
- [Instalación de Transformers y Datasets](#)
- [Importación de librerías](#)
- [Montado en /content/drive](#)
- [Enumerar los metadatos y el contenido del conjunto de datos](#)
- [Creación de un conjunto de entrenamiento y prueba a partir de un conjunto de datos](#)
- [Indexar y segmentar el conjunto de datos](#)
- [Cargar el modelo BERT previamente entrenado](#)
- [Tokenize the Data \(Tokenizar los datos\)](#)
- [Del conjunto train, obtenemos la pregunta, la respuesta a cada pregunta y la respuesta etiqueta dentro del contexto.](#)
- [Del conjunto validation, obtenemos la pregunta, la respuesta a cada pregunta y la respuesta etiqueta dentro del contexto.](#)
- [Vector de tokens de la pregunta y el contexto de todo el conjunto train.](#)
- [Vector de tokens de la pregunta y el contexto de todo el conjunto validation.](#)

- [Fine-Tuning del Modelo \(Ajustar el modelo\)](#)
- [Definir los Hiperparámetros \(argumentos de entrenamiento\)](#)
- [Congelar capas](#)
- [Agregar capas adicionales](#)
- [Aplicar LoRA al entrenamiento](#)
- [Primero, necesitamos definir una función de decaimiento de la tasa de aprendizaje. Para LoRA, la tasa de aprendizaje se ajusta linealmente en cada reinicio del entrenamiento. Se define una función para esto, que tomará como argumento el paso actual, el número total de pasos y la tasa de aprendizaje inicial.](#)
- [Luego, necesitamos crear un objeto de programador \(scheduler\) que aplicará esta función de decaimiento de la tasa de aprendizaje al optimizador durante el entrenamiento.](#)
- [Adam \(Adaptive Moment Estimation\)](#)
- [El scheduler LoRA con el optimizador SGD \(Stochastic Gradient Descent\)](#)
- [Finalmente, se pasa el scheduler junto con el optimizador al Trainer al crearlo.](#)
- [DefaultDataCollator](#)
- [Inicialización de Trainer con los argumentos y optimizadores y Entrenamiento](#)
- [Un valor de pérdida de entrenamiento de aproximadamente 0.5](#)
- [Método train\(\)](#)
- [Aumentar la retroalimentación](#)
- [Optimizador](#)
- [Guardar el modelo entrenado en disco](#)
- [Prueba con una instancia del conjunto de entrenamiento](#)
- [Tokenizar los datos de un ejemplo con el modelo](#)
- [Aplicar el modelo BERT a los datos de un ejemplo](#)
- [torch es una biblioteca de Python](#)
- [Selección del Dispositivo \(CPU o GPU\):](#)
- [Mover el Modelo al Dispositivo Seleccionado:](#)
- [Preparación de Entradas para el Modelo:](#)
- [Inferencia del Modelo BERT:](#)
- [Obtención de la Respuesta Predicha:](#)
- [Información Adicional:](#)
- [Evaluar el resultado de los datos de ejemplo con el modelo inicial](#)
- [Función para evaluar una sola instancia](#)
- [La librería tqdm Python](#)
- [Aplicar el modelo en un conjunto de instancias](#)

- [Cargar el modelo que guarde despues del entrenamiento](#)
- [Evaluar el resultado de los datos de ejemplo con el modelo 2 guardado y cargado](#)
- [Evaluación de las respuestas sobre el conjunto de datos de train](#)
- [Función para calcular las métricas de precisión, recall y F1-Score](#)
- [Evaluación de las respuestas se hace sobre el conjunto de datos de validación](#)
- [Función para calcular las métricas de precisión y recall con umbral](#)
- [Evaluación de las respuestas se hace sobre el conjunto de datos de validación. Precision y Recall con umbral.](#)
- [Tokenizar los datos de con el nuevo modelo fine-tuning](#)
- [Realizar una inferencia con el modelo entrenado con fine-tuning](#)
- [Hacer la inferencia sin contexto, recorriendo el dataset y buscar la pregunta \(question\) que mas se asemeja](#)
- [Contexto completo para inferencia](#)
- [pipeline\("question-answering"\)](#)
- [Inferencia con pipeline\("question-answering"\), para preguntas que entran en nuestro contexto](#)
- [Inferencia con pipeline\("question-answering"\), para preguntas que "No" entran en nuestro contexto](#)
- [Transformers de Hugging Face para realizar inferencias de preguntas y respuestas](#)
- [Modelo preentrenado completo para responder preguntas dadas un contexto de Avería y Contacto, sobre los servicios de agua en la empresa Canal de Isabel II de suministro de agua](#)
- [Modelo preentrenado reducido \(prueba\) para responder preguntas dadas un contexto específico reducido \(prueba\), sobre los servicios de agua en la empresa Canal de Isabel II de suministro de agua](#)

Autor: Antonio Fernández Salcedo / [Hugging Face](#)

Trabajando: Canal de Isabel II (<https://www.canaldeisabelsegunda.es>)

Formación: Grado Ingeniería Informática (<https://www.uoc.edu/es>)

Aplicación desarrollada para TFG. Inteligencia Artificial.

Contacto: afernandezsal@uoc.edu

Trabajar con conjuntos de datos de preguntas y respuestas etiquetadas y modelos de Hugging Face

Cargando y explorando un conjunto de datos de preguntas y respuestas

1. Cargaremos un conjunto de datos de preguntas y respuestas del Canal de Isabel II.
2. Extraeremos información del conjunto de datos y realizaremos un fine-tuning.

El método de fine-tuning se basará en:

1. Ajustar Hiperparametros esenciales.

`lr = 1e-5`. Tasa de aprendizaje: tamaño del paso de optimización (*size of optimization step*).

`batch_size = 8`. Tamaño del lote: número de ejemplos procesados por paso de optimización*(No.of.examples processed per optimization step).*

`num_epochs = 10`. Número de épocas: número de veces que el modelo atraviesa los datos de entrenamiento*(No.of.times the model runs through training data).*

`weight_decay = 0.006`. Decaimiento de peso para evitar sobreajuste.

2. Agregar capas adicionales que se integrarán en la arquitectura existente del modelo BERT para ajustarse mejor a la tarea específica de preguntas y respuestas.
3. *Linearly decaying Learning rate with Restart Algorithm (LoRA)* con el Optimizador SGD (*Stochastic Gradient Descent*).
4. *Linearly decaying Learning rate with Restart Algorithm (LoRA)* con el Optimizador Adam (*Adaptive Moment Estimation*).
5. *Linearly decaying Learning rate with Restart Algorithm (LoRA)* con el Optimizador AdamW (*Adam with Weight Decay*).

Requisitos de hardware

```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
    print('Not connected to a GPU')
else:
    print(gpu_info)
```

Tue May 14 14:55:01 2024

```

+-----+
| NVIDIA-SMI 535.104.05                Driver Version: 535.104.05   CUDA Version: 12.2   |
+-----+-----+-----+-----+-----+-----+
| GPU  Name   Persistence-M | Bus-Id  Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf    Pwr:Usage/Cap |          Memory-Usage | GPU-Util  Compute M. |
|                                           MIG M. |
+-----+-----+-----+-----+-----+-----+
|   0   Tesla T4            Off | 00000000:00:04:0  Off |                    0 |
| N/A   62C    P8             10W /  70W |  0MiB / 15360MiB |          0%      Default |
|                                           N/A   |
+-----+-----+-----+-----+-----+-----+

```

```
| Processes:
| GPU  GI  CI      PID  Type  Process name          GPU Memory
|      ID  ID                               Usage
|-----|-----|-----|-----|-----|-----|-----|
| No running processes found
|-----|-----|-----|-----|-----|-----|-----|
```

Aquí estamos usando una Tesla T4 GPU. Y soporta un entrenamiento de bastantes épocas.

Si `torch.cuda.is_available()` devuelve `False`, significa que no se detectó ninguna GPU disponible en tu sistema. Hay algunas razones comunes por las que esto podría suceder:

1. **Falta de GPU:** Si estás ejecutando el código en una máquina que no tiene una tarjeta gráfica compatible con CUDA, `torch.cuda.is_available()` devolverá `False`.
2. **Configuración incorrecta:** Asegúrate de que los controladores de la GPU estén instalados correctamente y que CUDA esté configurado adecuadamente en tu sistema.
3. **Entorno de ejecución:** Si estás ejecutando el código en un entorno como Google Colab o Kaggle, es posible que no se te asigne una GPU. Deberías verificar la configuración del entorno o intentar cambiar a uno que ofrezca aceleración por GPU.
4. **Limitaciones de acceso:** En algunos casos, el acceso a la GPU puede estar restringido debido a políticas de seguridad o privilegios de usuario. Asegúrate de tener los permisos adecuados para acceder a la GPU.

Si ninguna de estas soluciones resuelve el problema, puede haber problemas más específicos relacionados con la configuración del sistema.

```
import torch
torch.cuda.is_available()

True
```

RAM

```
from psutil import virtual_memory
ram_gb = virtual_memory().total / 1e9
print('El tiempo de ejecución tiene {:.1f} gigabytes de RAM disponible\n'.format(ram_gb))

if ram_gb < 12:
    print('Para habilitar un tiempo de ejecución de RAM alto, seleccione > tiempo de ejecución "Cambiar tipo de tiempo de ejecución')
else:
    print('¡Estas utilizando un tiempo de ejecución de alta RAM!')
```

El tiempo de ejecución tiene 13.6 gigabytes de RAM disponible

¡Estas utilizando un tiempo de ejecución de alta RAM!

Instalación de Transformers y Datasets

```
# Instalación
! pip install datasets transformers[torch]

Collecting datasets
  Downloading datasets-2.19.1-py3-none-any.whl (542 kB)
-----
                    542.0/542.0 kB 6.9 MB/s eta 0:00:00
Requirement already satisfied: transformers[torch] in /usr/local/lib/python3.10/dist-packages (4.40.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.14.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.25.2)
Requirement already satisfied: pyarrow=12.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (14.0.2)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-packages (from datasets) (0.6)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
-----
                    116.3/116.3 kB 7.3 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.4)
Collecting xxhash (from datasets)
  Downloading xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (194 kB)
-----
                    194.1/194.1 kB 5.1 MB/s eta 0:00:00
```

Importación de librerías

```
import json
from google.colab import drive
from datasets import Dataset
from datasets import DatasetDict
import pandas as pd
import numpy as np
from transformers import AutoTokenizer, AutoModelForQuestionAnswering
from transformers import TrainingArguments, Trainer
from transformers import DefaultDataCollator
from transformers import pipeline
from tqdm import tqdm
from IPython.core.display import HTML
from scipy.spatial.distance import cosine
```

Montado en /content/drive

Cargar un conjunto de datos de preguntas_respuestas.json específico del Canal de Isabel II

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
import json

# Abrir el archivo JSON en modo lectura
with open('/content/drive/MyDrive/TFG/preguntas_respuestas.json', 'r') as file:
    # Cargar el contenido del archivo JSON en una variable Python
    data = json.load(file)

# Ahora, la variable 'data' contiene los datos del archivo JSON
print(data)
```

```
[{'id': '100', 'title': 'Preguntas frecuentes: Averías', 'context': 'En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros.', 'question': '¿Qué debo hacer si no tengo agua en mi casa?', 'answers': {'text': ['En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua'], 'answer_start': [0]}}, {'id': '101', 'title': 'Preguntas frecuentes: Averías', 'context': 'En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros.', 'question': '¿Cuáles son las acciones recomendadas si carezco de agua en mi hogar?', 'answers': {'text': ['En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua'], 'answer_start': [0]}}, {'id': '102', 'title': 'Preguntas frecuentes: Averías', 'context': 'En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros.', 'question': '¿Cómo puedo encontrar la ubicación de mi contador?', 'answers': {'text': ['En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros.', 'question': '¿Cómo puedo encontrar la ubicación de mi contador?'], 'answer_start': [0]}]}
```

Enumerar los metadatos y el contenido del conjunto de datos

```
from datasets import Dataset

# Suponiendo que 'data' es la lista de datos que cargaste del archivo JSON
# Define las claves esperadas en el conjunto de datos
keys = ['id', 'title', 'context', 'question', 'answers']

# Convierte la lista de datos en un diccionario con las claves esperadas
data_dict = {key: [item[key] for item in data] for key in keys}
```

```
# Crea el conjunto de datos a partir del diccionario
dataset = Dataset.from_dict(data_dict)

# Imprime el conjunto de datos
print(dataset)

Dataset({
  features: ['id', 'title', 'context', 'question', 'answers'],
  num_rows: 506
})
```

```
from datasets import DatasetDict
```

```
# Convertir la lista de diccionarios en un DatasetDict
dataset_diccionario = DatasetDict({"train": dataset})

# Verificar el tipo de dataset
print(type(dataset_diccionario))
```

```
<class 'datasets.dataset_dict.DatasetDict'>
```

```
dataset_diccionario
```

```
DatasetDict({
  train: Dataset({
    features: ['id', 'title', 'context', 'question', 'answers'],
    num_rows: 506
  })
})
```

Creación de un conjunto de entrenamiento y prueba a partir de un conjunto de datos

```
# Dividir el conjunto de datos(diccionario) en conjuntos de datos de entrenamiento y validación
data = dataset_diccionario['train'].train_test_split(train_size=0.8, seed=42)
```

```
data
```

```
DatasetDict({
  train: Dataset({
    features: ['id', 'title', 'context', 'question', 'answers'],
    num_rows: 404
  })
  test: Dataset({
    features: ['id', 'title', 'context', 'question', 'answers'],
    num_rows: 102
  })
})
```

```
# Renombrar el valor predeterminado "test" por "val"
data['validation'] = data.pop("test")
```

```
data
```

```
DatasetDict({
  train: Dataset({
    features: ['id', 'title', 'context', 'question', 'answers'],
    num_rows: 404
  })
  validation: Dataset({
    features: ['id', 'title', 'context', 'question', 'answers'],
    num_rows: 102
  })
})
```

```
# Mostrar el dataset
dataset
```

```
Dataset({
  features: ['id', 'title', 'context', 'question', 'answers'],
  num_rows: 586
})
```

```
import pandas as pd
```

```
# Obtener el conjunto de datos
data2 = dataset
```

```
# Convertir a DataFrame
df2 = pd.DataFrame(data2)
```

```
# Visualizar el DataFrame
print(df2.head(19))
```

```

   id  Preguntas frecuentes: Averías \
0  100 Preguntas frecuentes: Averías
1  101 Preguntas frecuentes: Averías
2  102 Preguntas frecuentes: Averías
3  103 Preguntas frecuentes: Averías
4  104 Preguntas frecuentes: Averías
5  105 Preguntas frecuentes: Averías
6  106 Preguntas frecuentes: Averías
7  107 Preguntas frecuentes: Averías
8  108 Preguntas frecuentes: Averías
9  109 Preguntas frecuentes: Averías
10 110 Preguntas frecuentes: Averías
11 111 Preguntas frecuentes: Averías
12 112 Preguntas frecuentes: Averías
13 113 Preguntas frecuentes: Averías
14 114 Preguntas frecuentes: Averías
15 115 Preguntas frecuentes: Averías
16 116 Preguntas frecuentes: Averías
17 117 Preguntas frecuentes: Averías
18 118 Preguntas frecuentes: Averías

                                     context \
0  En primer lugar, averigua si la falta de agua ...
1  En primer lugar, averigua si la falta de agua ...
2  En primer lugar, averigua si la falta de agua ...
3  En primer lugar, averigua si la falta de agua ...
4  En primer lugar, averigua si la falta de agua ...
5  En primer lugar, averigua si la falta de agua ...
6  En primer lugar, averigua si la falta de agua ...
7  En primer lugar, averigua si la falta de agua ...
8  En primer lugar, averigua si la falta de agua ...
9  En primer lugar, averigua si la falta de agua ...
10 En primer lugar, averigua si la falta de agua ...
11 En primer lugar, averigua si la falta de agua ...
12 En primer lugar, averigua si la falta de agua ...
13 En primer lugar, averigua si la falta de agua ...
14 En primer lugar, averigua si la falta de agua ...
15 En primer lugar, averigua si la falta de agua ...
16 En primer lugar, averigua si la falta de agua ...
17 En primer lugar, averigua si la falta de agua ...
18 En primer lugar, averigua si la falta de agua ...

                                     question \
0      ¿Qué debo hacer si no tengo agua en mi casa?
1  ¿Cuáles son las acciones recomendadas si carez...
2  ¿Qué pasos debo seguir si no tengo suministro ...
3  Si no tengo acceso a agua en mi vivienda, ¿qué...
4  ¿Cómo debo actuar si enfrente una situación en...
5  ¿Qué debo revisar si tengo falta de agua en mi...
6  ¿Qué debo verificar si no tengo agua en mi casa?
7  Si carezco de agua en mi vivienda, ¿qué elemen...
```

```
df2.head(3)
```

	id	title	context	question	answers
0	100	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Qué debo hacer si no tengo agua en mi casa?	('answer_start': [0], 'text': ['En primer luga...
1	101	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Cuáles son las acciones recomendadas si carez...	('answer_start': [0], 'text': ['En primer luga...
2	102	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Qué pasos debo seguir si no tengo suministro ...	('answer_start': [0], 'text': ['En primer luga...

Indexar y segmentar el conjunto de datos

```
# Consigue el registro 1 del conjunto de datos
dataset[1]
```

```
{'id': '101',
 'title': 'Preguntas frecuentes: Averías',
 'context': 'En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros.',
 'question': '¿Cuáles son las acciones recomendadas si carezco de agua en mi hogar?',
 'answers': {'answer_start': [0],
 'text': ['En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua']}]}
```

```
# Obtener los primeros 13 registros del conjunto de datos
test_last = dataset[0:12]
```

```
# Crear un DataFrame de Pandas
pd.DataFrame(test_last)
```

	id	title	context	question	answers
0	100	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Qué debo hacer si no tengo agua en mi casa?	('answer_start': [0], 'text': ['En primer luga...
1	101	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Cuáles son las acciones recomendadas si carez...	('answer_start': [0], 'text': ['En primer luga...
2	102	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Qué pasos debo seguir si no tengo suministro ...	('answer_start': [0], 'text': ['En primer luga...
3	103	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	Si no tengo acceso a agua en mi vivienda, ¿qué...	('answer_start': [0], 'text': ['En primer luga...
4	104	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Cómo debo actuar si enfrente una situación en...	('answer_start': [0], 'text': ['En primer luga...
5	105	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Qué debo revisar si tengo falta de agua en mi...	('answer_start': [163], 'text': ['Comprueba qu...
6	106	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Qué debo verificar si no tengo agua en mi casa?	('answer_start': [163], 'text': ['Comprueba qu...
7	107	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	Si carezco de agua en mi vivienda, ¿qué elemen...	('answer_start': [163], 'text': ['Comprueba qu...
8	108	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	Si enfrente escasez de agua en mi hogar, ¿qué ...	('answer_start': [163], 'text': ['Comprueba qu...
9	109	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Cuáles son las cosas básicas que debo inspec...	('answer_start': [163], 'text': ['Comprueba qu...
10	110	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Dónde puedo encontrar la ubicación del contador?	('answer_start': [384], 'text': ['En la bateri...
11	111	Preguntas frecuentes: Averías	En primer lugar, averigua si la falta de agua ...	¿Dónde debo buscar la ubicación del contador?	('answer_start': [384], 'text': ['En la bateri...

```
# Acceder a Los primeros 5 ejemplos de el conjunto dedel diccionario
data_train = data['train'][:5]
```

```
# Imprimir los primeros 5 ejemplos del conjunto de datos
for example in data_train:
    print(example, ":", data_train[example])
```

```
id : ['333', '1052', '711', '311', '933']
title : ['Bonificaciones', 'Contratación y cambio de titularidad', 'Gestiones y contratos', 'Bonificaciones', 'Facturación']
context : ['Pensión de viudedad. Perceptores de una pensión por viudedad con renta total inferior a 14.000 euros anuales. La b
onificación será del 50 % del importe de la cuota de servicio fija. Documentación que debes aportar por pensión de viudedad: Ce
rtificado, emitido por los órganos o entidades competentes, que acredite la condición de perceptor de una pensión de viudedad y
el importe de la misma. Fotocopia del Documento Nacional de Identidad del solicitante. Solicitud de bonificación de la factura
de consumo de agua, marcando la casilla de declaración responsable de que sus ingresos totales, incluida la pensión de viueda
d, no superan el importe establecido.', 'El Decreto 3068/75, de 31 de octubre, establece que en la contratación del suministro
se pueden facturar los siguientes conceptos: 1. Cuota de red: corresponde al importe de las obras de adaptación y mejora de la
red existente. Este concepto se facturará como norma general para los nuevos suministros solicitados. 2. Cuota de enganche: se
trata de los costes derivados de la ejecución de la obra, tales como materiales y mano de obra. 3. Anticipo de consumo: es una
cantidad que cubre el importe de la facturación del suministro de agua y que responde a las obligaciones económicas contraídas.
Este importe será devuelto al cliente a la finalización de la relación contractual, mediante la solicitud expresa del mismo y s
iempre y cuando se hayan cumplido las obligaciones contraídas por parte de este. Este concepto no existe en el caso de aconetid
as de protección contra incendios (PCI) y en los distribuidores principales (DP) y distribuidores principales únicos (DPU) que
dan servicio a baterías de contadores.', 'Si no eres usuario de la Oficina virtual puedes realizar las siguientes gestiones: 1.
Relacionadas con contratación: Contratar un suministro, darte de alta como cliente sensible y realizar gestiones asociadas con
alcantarillado y saneamiento. 2.Relacionadas con tus facturas: Pagar tus facturas, visualizar facturas electrónicas, realizar s
imulaciones de facturas, solicitar bonificaciones y darte de alta en la factura electrónica. 3.Relativas a consumos y lecturas:
Consultar la fecha de la próxima lectura y enviar la lectura de tu contador. 4.Otro tipo de gestiones: Cambiar la titularidad d
e un contrato, darte de alta en la Oficina virtual, pedir una cita para atención personalizada en una oficina, gestionar tus ci
tas, comunicar incidencias, consultar solicitudes, acceder a la oficina de reclamaciones y notificar incidencias en el suminst
ro.', 'Puede beneficiarse de las bonificaciones del Canal de Isabel II: Perceptores de una pensión por viudedad. Familias o viv
iendas numerosas. Entidades sin ánimo de lucro titulares de viviendas comunitarias o pisos tutelados. Beneficiarios de una pens
ión no contributiva. Beneficiarios de la renta mínima de inserción Beneficiarios de la renta activa de inserción. Perceptores d
el ingreso mínimo vital. Personas en situación de especial exclusión que cuenten con un informe del trabajador social. Personas
con vivienda legítima en situación de ocupación ilegal.', 'Según establece el Reglamento para el Servicio y Distribución de las
Aguas de Canal de Isabel II, (Decreto 2922/1975, de 31 de octubre) podríamos suspender el suministro de agua si transcurridos 3
0 días naturales desde la emisión de la factura, esta no ha sido abonada. El corte del suministro implicaría continuar facturand
o las cuotas de servicio. El restablecimiento del servicio se realizará una vez liquidada la deuda, así como el importe del re
stablecimiento. Transcurridos tres meses sin que se haya producido el pago, Canal puede resolver el contrato y proceder a la co
ndena de la acometida (retirada de la instalación). Para volver a tener agua habrá que realizar una nueva contratación.']
question : ['¿Cuáles son los grupos que reciben beneficios en la pensión de viudedad según el Canal de Isabel II?', '¿Qué gasto
s incluye la contratación de un nuevo suministro de agua?', '¿Qué opciones puedo llevar a cabo en relación con mis facturas si
no soy usuario de la Oficina Virtual?', '¿Quiénes son elegibles para recibir beneficios del Canal de Isabel II?', '¿En qué caso
s puede Canal de Isabel II cortar el suministro de agua?']
answers : [{'answer_start': [22], 'text': ['Perceptores de una pensión por viudedad con renta total inferior a 14.000 euros anu
ales']}, {'answer_start': [131], 'text': ['1. Cuota de red: corresponde al importe de las obras de adaptación y mejora de la re
d existente. Este concepto se facturará como norma general para los nuevos suministros solicitados. 2. Cuota de enganche: se tr
ata de los costes derivados de la ejecución de la obra, tales como materiales y mano de obra. 3. Anticipo de consumo: es una ca
ntidad que cubre el importe de la facturación del suministro de agua']}, {'answer_start': [275], 'text': ['Pagar tus facturas,
visualizar facturas electrónicas, realizar simulaciones de facturas, solicitar bonificaciones y darte de alta en la factura ele
ctrónica']}, {'answer_start': [65], 'text': ['Perceptores de una pensión por viudedad. Familias o viviendas numerosas']}, {'ans
wer_start': [136], 'text': ['Podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de l
a factura, esta no ha sido abonada']]}
```

Cargar el modelo BERT previamente entrenado

Para las tareas de respuesta a preguntas, un modelo previamente entrenado comúnmente utilizado es BERT (*Bidirectional Encoder Representations from Transformer*/Representaciones de codificador bidireccional de Transformers).

Hugging Face proporciona una interfaz fácil de usar para cargar este modelo junto con su tokenizador.

```
from transformers import AutoTokenizer, AutoModelForQuestionAnswering

#Cargar el tokenizador y el modelo
tokenizer = AutoTokenizer.from_pretrained('dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-miqa')
tokenizer
```

```
vocab.txt: 0%|          | 0.00/248k [00:00<?, ?B/s]
tokenizer.json: 0%|          | 0.00/486k [00:00<?, ?B/s]
special_tokens_map.json: 0%|          | 0.00/112 [00:00<?, ?B/s]

BertTokenizerFast(name_or_path='dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa', vocab_size=31002, model_max_length=512, is_fast=True, padding_side='right', truncation_side='right', special_tokens={'unk_token': '[UNK]', 'sep_token': '[SEP]', 'pad_token': '[PAD]', 'cls_token': '[CLS]', 'mask_token': '[MASK]'}, clean_up_tokenization_spaces=True), added_tokens_decoder={
  0: AddedToken("[MASK]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  1: AddedToken("[PAD]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  3: AddedToken("[UNK]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  4: AddedToken("[CLS]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  5: AddedToken("[SEP]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
}
```

Los datos proporcionados por el tokenizador tienen el siguiente significado:

1. `name_or_path`: Es la ruta o el nombre del modelo que el tokenizador está utilizando o cargando. En este caso, parece ser la ruta de un modelo almacenado en Google Drive.
2. `vocab_size`: Es el tamaño del vocabulario del tokenizador, es decir, el número total de tokens únicos que el tokenizador puede manejar.
3. `model_max_length`: Es la longitud máxima de secuencia admitida por el modelo. Para secuencias más largas, el tokenizador truncará o dividirá la secuencia de entrada según sea necesario.
4. `is_fast`: Indica si el tokenizador es una versión rápida (True) o no (False).
5. `padding_side`: Indica el lado del texto al que se aplicará el relleno. En este caso, el relleno se aplica en el lado derecho.
6. `truncation_side`: Indica el lado del texto al que se aplicará la truncación si la secuencia excede `model_max_length`. En este caso, la truncación se aplica en el lado derecho.
7. `special_tokens`: Es un diccionario que contiene tokens especiales y sus representaciones específicas. Estos tokens especiales se utilizan para indicar el comienzo y el final de una secuencia, así como para el relleno, el enmascaramiento y tokens desconocidos. Por ejemplo, `[PAD]` se utiliza para el relleno, `[CLS]` para el inicio de la secuencia, `[SEP]` para separar secuencias y `[MASK]` para el enmascaramiento.
8. `added_tokens_decoder`: Es un diccionario que contiene tokens adicionales agregados al vocabulario original del modelo. Cada token tiene un índice asignado y sus atributos correspondientes, como si es especial, si es un token único o si necesita ser normalizado.

```
# Cargar el modelo
model = AutoModelForQuestionAnswering.from_pretrained('dccuchile/bert-base-spanish-wwm-uncased-finetuned-qa-mlqa')
model
```

```
config.json: 0%|          | 0.00/718 [00:00<?, ?B/s]
pytorch_model.bin: 0%|          | 0.00/437M [00:00<?, ?B/s]

BertForQuestionAnswering(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(31002, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
        )
      )
      (intermediate): BertIntermediate(
        (dense): Linear(in_features=768, out_features=3072, bias=True)
        (intermediate_act_fn): GELUActivation()
      )
    )
  )
)
```

Tokenize the Data (Tokenizar los datos)

Los datos del dataset deben tokenizarse utilizando el tokenizador del modelo seleccionado.

La función `prepare_train_features` toma como entrada un conjunto de ejemplos de entrenamiento (diccionarios) y los transforma en un conjunto de características tokenizadas y etiquetadas para entrenar un modelo de respuesta a preguntas (QA). Las características tokenizadas se basan en el modelo de lenguaje pre-entrenado `tokenizer`. Los pasos que realiza la función:

1. Tokenización y truncamiento:

- La función tokeniza la pregunta y el contexto de cada ejemplo utilizando el `tokenizer` pre-entrenado.
- Se aplica un truncamiento "solo al segundo" para mantener la pregunta completa y trunca el contexto si es necesario.
- La longitud máxima de la secuencia se establece en 512 tokens.
- Se utiliza un "paso" de 128 para generar múltiples características a partir de un contexto largo, superponiendo un poco cada característica.

2. Mapeo de características y offsets:

- Se crea un mapeo de "desbordamiento a muestra" para identificar el ejemplo original de cada característica generada a partir de un contexto largo.
- Se crea un mapeo de "offsets" para obtener la posición del carácter original para cada token en la secuencia tokenizada.

3. Etiquetado de ejemplos:

- Se inicializan las listas `start_positions` y `end_positions` para almacenar las posiciones de inicio y fin de las respuestas en la secuencia tokenizada.
- Para cada ejemplo:

- Se verifica si hay respuestas en el ejemplo.

- Si no hay respuestas, se establece la respuesta como el token CLS (índice especial).
- Si hay respuestas:

a. Se obtienen las posiciones de inicio y fin del carácter de la respuesta en el texto original.

b. Se calcula el índice de inicio y fin del token de la respuesta en la secuencia tokenizada.

c. Se comprueba si la respuesta está dentro del fragmento actual.

d. Si la respuesta está fuera del fragmento, se establece la respuesta como el token CLS.

e. Si la respuesta está dentro del fragmento, se ajustan los índices de inicio y fin del token para abarcar la respuesta.

4. Salida:

La función devuelve un diccionario con las características tokenizadas y etiquetadas:

1. `input_ids`: codificación numérica de los tokens `context + question`
2. `token_type_ids`: secuencia (del mismo tamaño de `input_ids`) indicando la porción correspondiente a `context` (codificada con 1s), `answer` (codificada con 0s) y "padding" (también con 0s)
3. `attention_mask`: la máscara atencional (del mismo tamaño de `input_ids`) indicando la porción correspondiente a `context + answer` (codificada con 1s) y la porción que contiene "padding" (codificada con 0s)
4. `start_positions`: la posición del token de inicio de la respuesta, con respecto a la secuencia `input_ids`
5. `end_positions`: la posición del token de finalización de la respuesta, con respecto a la secuencia `input_ids`

En resumen, la función `prepare_train_features` transforma ejemplos de entrenamiento en características tokenizadas y etiquetadas que se pueden utilizar para entrenar un modelo de respuesta a preguntas.

Aquí hay algunos puntos adicionales a tener en cuenta:

- La función está diseñada para trabajar con el formato de datos específico del conjunto de datos utilizado.
- La característica `padding` en la función de tokenización: En la función `prepare_train_features`, se ha definido `padding="max_length"`. Esto indica que todas las secuencias de entrada (tanto las preguntas como los contextos) serán rellenadas con tokens de padding para que tengan la misma longitud máxima (`max_length`). Este es un paso importante para asegurarse de que todas las secuencias en un lote tengan la misma longitud.

```

# Definir La función de tokenización y procesamiento de Los datos
def prepare_train_features(examples):
    # Tokenizamos nuestros ejemplos con truncamiento y padding, pero mantenemos Los desbordamientos usando un paso.
    # Esto resulta en que un ejemplo pueda dar varias características cuando un contexto es largo,
    # cada una de esas características con un contexto que se superpone un poco al contexto de La característica anterior.
    tokenized_examples = tokenizer(
        examples["question"],
        examples["context"],
        truncation="only_second", # Truncar contexto, no La pregunta
        max_length=512,          # Longitud máxima de La secuencia
        stride=128,
        return_overflowing_tokens=True,
        return_offsets_mapping=True,
        padding="max_length",
    )

    # Dado que un ejemplo podría darnos varias características si tiene un contexto largo,
    # necesitamos un mapeo de una característica a su ejemplo correspondiente. Esta clave nos da justo eso.
    sample_mapping = tokenized_examples.pop("overflow_to_sample_mapping")
    # Los mapeos de desplazamiento nos darán un mapa desde el token hasta La posición del carácter en el contexto original.
    # Esto nos ayudará a calcular Las posiciones de inicio y fin.
    offset_mapping = tokenized_examples.pop("offset_mapping")

    # Etiquetemos esos ejemplos
    tokenized_examples["start_positions"] = []
    tokenized_examples["end_positions"] = []

    for i, offsets in enumerate(offset_mapping):
        # Etiquetaremos respuestas imposibles con el índice del token CLS.
        input_ids = tokenized_examples["input_ids"][i]
        cls_index = input_ids.index(tokenizer.cls_token_id)

        # Capturamos La secuencia correspondiente a ese ejemplo (para saber qué es el contexto y qué es La pregunta).
        sequence_ids = tokenized_examples.sequence_ids(i)

        # Un ejemplo puede dar varios fragmentos, este es el índice del ejemplo que contiene este fragmento de texto.
        sample_index = sample_mapping[i]
        answers = examples["answers"][sample_index]
        # Si no se dan respuestas, establecer el índice cls como respuesta.
        if len(answers["answer_start"]) == 0:
            tokenized_examples["start_positions"].append(cls_index)
            tokenized_examples["end_positions"].append(cls_index)
        else:
            # Índice de inicio/fin de caracteres de La respuesta en el texto.
            start_char = answers["answer_start"][0]
            end_char = start_char + len(answers["text"][0])

            # Índice de inicio del token del fragmento actual en el texto.
            token_start_index = 0
            while sequence_ids[token_start_index] != 1:
                token_start_index += 1

            # Índice de fin del token del fragmento actual en el texto.
            token_end_index = len(input_ids) - 1
            while sequence_ids[token_end_index] != 1:
                token_end_index -= 1

            # Detectar si La respuesta está fuera del fragmento (en cuyo caso esta característica se etiqueta con el índice CLS)
            if not (offsets[token_start_index][0] <= start_char and offsets[token_end_index][1] >= end_char):
                tokenized_examples["start_positions"].append(cls_index)
                tokenized_examples["end_positions"].append(cls_index)
            else:
                # De Lo contrario, mover el índice de inicio del token y el índice de fin del token a Los dos extremos de La res
                # Nota: podríamos ir después del último desplazamiento si La respuesta es La última palabra (caso límite).
                while token_start_index < len(offsets) and offsets[token_start_index][0] <= start_char:
                    token_start_index += 1
                tokenized_examples["start_positions"].append(token_start_index - 1)
                while offsets[token_end_index][1] >= end_char:
                    token_end_index -= 1
                tokenized_examples["end_positions"].append(token_end_index + 1)

    return tokenized_examples

```

```
example = data['train'][8]
for key in example:
    print(key, ":", example[key])
```

```
id : 948
title : Facturación
context : Según establece el Reglamento para el Servicio y Distribución de las Aguas de Canal de Isabel II, (Decreto 2922/1975, de 31 de octubre) podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de la factura, esta no ha sido abonada. El corte del suministro implicaría continuar facturando las cuotas de servicio. El restablecimiento de l servicio se realizará una vez liquidada la deuda, así como el importe del restablecimiento. Transcurridos tres meses sin que se haya producido el pago, Canal puede resolver el contrato y proceder a la condena de la acometida (retirada de la instalación). Para volver a tener agua habrá que realizar una nueva contratación.
question : ¿Qué implicaría el corte del suministro de agua?
answers : {'answer_start': [270], 'text': ['El corte del suministro implicaría continuar facturando las cuotas de servicio']}
```

```
# Aplicar la función a los datos
tokenized_datasets = data.map(prepare_train_features, batched=True, remove_columns=data["train"].column_names)
```

```
Map: 0% | 0/404 [00:00<?, ? examples/s]
```

```
Map: 0% | 0/102 [00:00<?, ? examples/s]
```

```
tokenized_datasets
```

```
DatasetDict({
  train: Dataset({
    features: ['input_ids', 'token_type_ids', 'attention_mask', 'start_positions', 'end_positions'],
    num_rows: 404
  })
  validation: Dataset({
    features: ['input_ids', 'token_type_ids', 'attention_mask', 'start_positions', 'end_positions'],
    num_rows: 102
  })
})
```

Cuando tokenizamos el dataset, obtenemos varios vectores que representan la información del texto en un formato adecuado. El significado de cada vector:

1. input_ids:

- Contiene los identificadores (IDs) enteros asignados a cada token en la secuencia de entrada después de la tokenización.
- Cada token en el texto original se mapea a un ID único basado en el vocabulario del modelo (un diccionario que vincula palabras a números).
- Este vector tiene la misma longitud que la secuencia tokenizada.
- Ejemplo: Si la tercera palabra en tu texto original es "Antonio" y su ID en el vocabulario es 24, entonces `input_ids[2]` sería 24.

2. token_type_ids:

- Este vector es opcional y depende del modelo y la tarea específica.
- Se utiliza en algunos modelos de Transformers para diferenciar entre tokens de diferentes segmentos en la entrada.
- Por ejemplo, en nuestro caso [CLS] (indicando clasificación) al principio y [SEP] (indicando separación) entre las oraciones".

3. attention_mask:

- Este vector es un vector binario (contiene 0s y 1s) de la misma longitud que `input_ids`.
- Indica al modelo qué partes de la secuencia de entrada son relevantes para la tarea y cuáles se deben ignorar.
- Los valores 1 corresponden a tokens válidos que el modelo debe atender, mientras que los valores 0 corresponden a tokens que se deben ignorar (por ejemplo, *padding*).
- Se utiliza para manejar secuencias de diferentes longitudes y asegurarse de que el modelo no preste atención a tokens de relleno.

4. start_positions:

- Este vector es opcional y depende de la tarea específica (generalmente, tareas de respuesta a preguntas).
- Contiene el índice del primer token en la secuencia de entrada que corresponde al inicio de la respuesta prevista.
- Si no estás realizando una tarea de respuesta a preguntas, es posible que este vector no esté presente.

5. end_positions:

- Este vector es opcional y depende de la tarea específica (generalmente, tareas de respuesta a preguntas).
- Contiene el índice del último token en la secuencia de entrada que corresponde al final de la respuesta prevista.
- Si no estás realizando una tarea de respuesta a preguntas, es posible que este vector no esté presente.

Del conjunto train, obtenemos la pregunta, la respuesta a cada pregunta y la respuesta etiqueta dentro del contexto.

```
row = 1
for example in data['train']:
    print(f"{row}: {example['question']}")
    row += 1
```

```
1: ¿Cuáles son los grupos que reciben beneficios en la pensión de viudedad según el Canal de Isabel II?
2: ¿Qué gastos incluye la contratación de un nuevo suministro de agua?
3: ¿Qué opciones puedo llevar a cabo en relación con mis facturas si no soy usuario de la Oficina Virtual?
4: ¿Quiénes son elegibles para recibir beneficios del Canal de Isabel II?
5: ¿En qué casos puede Canal de Isabel II cortar el suministro de agua?
6: ¿Qué pasos debo seguir para aumentar el caudal de agua en mi vivienda?
7: ¿Qué pasa si no cambio el nombre del contrato del agua después de comprar una casa?
8: ¿Qué pasos debo seguir si no tengo suministro de agua en casa?
9: ¿Qué implicaría el corte del suministro de agua?
10: ¿En qué consiste la cuota de servicio en la factura?
11: ¿Cómo se compone la factura del agua?
12: ¿Qué información debes proporcionar en todas las comunicaciones?
13: ¿Hay alguna app para saber si no va a haber agua?
14: ¿Dónde puedo encontrar el servicio de notificaciones de incidencias en el suministro de agua?
15: ¿Poseen una dirección de correo electrónico a la que pueda enviar un mensaje?
16: ¿A quiénes se les aplica la bonificación por exención social en el Canal de Isabel II?
17: ¿Existe la posibilidad de pagar en cuotas la factura del agua en Canal de Isabel II?
18: ¿Cómo puedo remitir la lectura del contador a Canal de Isabel II?
19: ¿Por qué me piden la licencia de primera ocupación para darme de alta el agua?
```

```
row = 1
for example in data['train']:
    print(f"{row}: {example['answers']['text'][0]}")
    start = example['answers']['answer_start'][0]
    print(f" {example['context'][:start:(start + len(example['answers']['text'][0]))]}")
    row += 1
```

```
1: Perceptores de una pensión por viudedad con renta total inferior a 14.000 euros anuales
   Perceptores de una pensión por viudedad con renta total inferior a 14.000 euros anuales
2: 1. Cuota de red: corresponde al importe de las obras de adaptación y mejora de la red existente. Este concepto se factura
   rá como norma general para los nuevos suministros solicitados. 2. Cuota de enganche: se trata de los costes derivados de la
   ejecución de la obra, tales como materiales y mano de obra. 3. Anticipo de consumo: es una cantidad que cubre el importe de
   la facturación del suministro de agua
   1. Cuota de red: corresponde al importe de las obras de adaptación y mejora de la red existente. Este concepto se facturar
   á como norma general para los nuevos suministros solicitados. 2. Cuota de enganche: se trata de los costes derivados de la e
   jecución de la obra, tales como materiales y mano de obra. 3. Anticipo de consumo: es una cantidad que cubre el importe de l
   a facturación del suministro de agua
3: Pagar tus facturas, visualizar facturas electrónicas, realizar simulaciones de facturas, solicitar bonificaciones y darte
   de alta en la factura electrónica
   Pagar tus facturas, visualizar facturas electrónicas, realizar simulaciones de facturas, solicitar bonificaciones y darte
   de alta en la factura electrónica
4: Perceptores de una pensión por viudedad. Familias o viviendas numerosas
   Perceptores de una pensión por viudedad. Familias o viviendas numerosas
5: Podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de la factura, esta no ha s
   ido abonada
   podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de la factura, esta no ha si
```

```
row = 1
for example in tokenized_datasets['train']:
    answer = tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(example['input_ids'][example['start_positions']:ex
    print(f"{row}: {answer}")
    row += 1
```

```
1: answer = perceptores de una pensión por viudedad con renta total inferior a 14. 000 euros anuales
2: answer = 1. cuota de red : corresponde al importe de las obras de adaptación y mejora de la red existente. este concepto
   se facturará como norma general para los nuevos suministros solicitados. 2. cuota de enganche : se trata de los costes deriv
   ados de la ejecución de la obra, tales como materiales y mano de obra. 3. anticipo de consumo : es una cantidad que cubre el
   importe de la facturación del suministro de agua
3: answer = pagar tus facturas, visualizar facturas electrónicas, realizar simulaciones de facturas, solicitar bonificacion
   es y darte de alta en la factura electrónica
4: answer = perceptores de una pensión por viudedad. familias o viviendas numerosas
5: answer = podríamos suspender el suministro de agua si transcurridos 30 días naturales desde la emisión de la factura, es
   ta no ha sido abonada
```


Fine-Tuning del Modelo (Ajustar el modelo)

Configuramos los argumentos para el entrenamiento del modelo utilizando la biblioteca Transformers. Estos argumentos incluyen la estrategia de evaluación, la tasa de aprendizaje, el tamaño del lote de entrenamiento y evaluación, el número de épocas de entrenamiento y el decaimiento de peso. Estos parámetros son esenciales para controlar cómo se realizará el entrenamiento del modelo.

Para asegurarse de que se registre la pérdida de entrenamiento correctamente, agregamos el argumento `report_to` en los `TrainingArguments` y se establece en `"all"` para que se informen todos los resultados, incluida la pérdida de entrenamiento.

Definir los Hiperparámetros (argumentos de entrenamiento)

Notación científica

- "1e-3" es una notación abreviada para representar un número en notación científica. En este caso, "1e-3" significa "1 multiplicado por 10 elevado a la potencia de -3". Simplificando, esto se traduce a "0.001".
- Entonces, "1e-3" es equivalente a "0.001". Esta notación se utiliza comúnmente en programación y ciencias para representar números pequeños de manera más concisa. En este caso específico, "1e-3" representa el número decimal "0.001".
- El número "2e-05" es una notación abreviada para representar un número en notación científica. En este caso, "2e-05" significa "2 multiplicado por 10 elevado a la potencia de -5". Simplificando, esto se traduce a "0.00002".
- Entonces, "2e-05" es equivalente a "0.00002". Esta notación se utiliza comúnmente en la programación y en disciplinas científicas para representar números muy pequeños de manera más concisa.

```
# Hiperparámetros
lr = 1e-5           # Tasa de aprendizaje: tamaño del paso de optimización
batch_size = 8     # Tamaño del lote: número de ejemplos procesados por paso de optimización
num_epochs = 10    # Número de épocas: número de veces que el modelo atraviesa los datos de entrenamiento
weight_decay = 0.006 # Decaimiento de peso para evitar el sobreajuste
```

Explicación detallada de cada argumento utilizado en la definición de los argumentos de entrenamiento para el fine-tuning del modelo preentrenado de BERT:

1. `output_dir='results'`: Este argumento especifica el directorio donde se guardarán los resultados del entrenamiento, como los modelos guardados y otros archivos relacionados con el entrenamiento.
2. `num_train_epochs=num_epochs`: Indica el número de épocas de entrenamiento que se realizarán. Cada época representa un pase completo a través de todo el conjunto de datos de entrenamiento.
3. `per_device_train_batch_size=batch_size`: Determina el tamaño del lote de entrenamiento por dispositivo. Especifica cuántos ejemplos se procesarán en paralelo en cada iteración de entrenamiento en cada dispositivo.
4. `per_device_eval_batch_size=batch_size`: Similar al argumento anterior, pero para el tamaño del lote durante la evaluación del modelo.
5. `evaluation_strategy='epoch'`: Especifica la estrategia de evaluación, en este caso, se evaluará el modelo al final de cada época.
6. `logging_dir='logs'`: Define el directorio donde se guardarán los registros de entrenamiento, como los registros de pérdida y métricas durante el entrenamiento.
7. `logging_steps=100`: Establece la frecuencia con la que se guardarán los registros durante el entrenamiento. En este caso, se guardarán registros cada 100 pasos de entrenamiento.
8. `save_steps=1000`: Define cada cuántos pasos se guardará el modelo durante el entrenamiento. Aquí, se guardará el modelo cada 1000 pasos.
9. `save_total_limit=3`: Limita el número total de modelos guardados. En este caso, se guardarán hasta 3 modelos.
10. `learning_rate=lr`: Especifica la tasa de aprendizaje inicial para el optimizador. Controla la magnitud de los ajustes realizados a los pesos del modelo durante el entrenamiento.
11. `weight_decay=0.01`: Define el factor de decaimiento de peso para evitar el sobreajuste. Ayuda a regularizar el modelo durante el entrenamiento.
12. `report_to="all"`: Indica que se deben reportar todos los resultados, incluida la pérdida de entrenamiento, en los registros y otras salidas.
13. `save_strategy="epoch"`: Determina la estrategia de guardado de modelos. En este caso, se guarda el modelo al final de cada época.
14. `load_best_model_at_end=True`: Indica que se debe cargar el mejor modelo al final del entrenamiento. Esto ayuda a asegurar que el modelo final sea el mejor en términos de métricas de evaluación.

```

from transformers import TrainingArguments, Trainer

# Definir Los argumentos de entrenamiento
training_args = TrainingArguments(
    output_dir='./results4',           # Directorio donde se guardarán Los resultados del entrenamiento
    num_train_epochs=num_epochs,      # Número de épocas de entrenamiento
    per_device_train_batch_size=batch_size, # Tamaño del Lote de entrenamiento por dispositivo
    per_device_eval_batch_size=batch_size, # Tamaño del Lote de evaluación por dispositivo
    evaluation_strategy='epoch',      # Estrategia de evaluación
    logging_dir='./logs4',           # Directorio donde se guardarán Los Logs de entrenamiento
    logging_steps=100,               # Cada cuántos pasos se guardarán Los Logs
    save_steps=1000,                 # Cada cuántos pasos se guardará el modelo
    save_total_limit=3,              # Límite total de modelos guardados
    learning_rate=lr,                # Tasa de aprendizaje inicial para el optimizador
    weight_decay=weight_decay,       # Decaimiento de peso para evitar sobreajuste
    report_to="all",                 # Reportar todos Los resultados, incluida La pérdida de entrenamiento
    save_strategy="epoch",           # Estrategia de guardado del modelo
    load_best_model_at_end=True,     # Cargar el mejor modelo al final del entrenamiento
)

```

Revisando los `TrainingArguments`, vemos todas las configuraciones relacionadas con el entrenamiento del modelo, desde el directorio donde se guardarán los resultados (`output_dir`) hasta la tasa de aprendizaje (`learning_rate`), la estrategia de evaluación (`evaluation_strategy`), la estrategia de guardado del modelo (`save_strategy`), entre otros.

Observando el parámetro `optim`, el valor es `adamw_torch`, lo que indica que se está utilizando el optimizador AdamW de PyTorch.

La configuración del optimizador se realiza a través del parámetro `learning_rate`, que indica la tasa de aprendizaje inicial para el optimizador.

La biblioteca transformers suele utilizar optimizadores como AdamW (una variante de Adam con la corrección de peso de decaimiento) de forma predeterminada, a menos que especifiques un optimizador diferente explícitamente.

La diferencia entre Adam y AdamW radica en cómo manejan el término de decaimiento de peso (`weight_decay`) durante el entrenamiento.

1. Adam (Adaptive Moment Estimation):

- Adam es un optimizador popular utilizado en el entrenamiento de redes neuronales. Combina las ventajas de RMSProp y Momentum.
- En Adam, el término de decaimiento de peso se aplica directamente a los pesos durante la actualización. Sin embargo, esto puede causar problemas de regularización y sobreajuste.

2. AdamW (Adam with Weight Decay):

- AdamW es una variante de Adam que aborda el problema de regularización.
- En AdamW, el término de decaimiento de peso se aplica de manera diferente. Utiliza lo que se llama "Decoupled Weight Decay Regularization". Esto significa que el decaimiento de peso no afecta directamente los pesos durante la actualización, sino que se aplica por separado después de la actualización.
- La implementación de AdamW asegura que el decaimiento de peso no interfiera con los momentos acumulados, lo que ayuda a prevenir el sobreajuste.

```

training_args
TrainingArguments(
  _n_gpu=1,
  accelerator_config={'split_batches': False, 'dispatch_batches': None, 'even_batches': True, 'use_seedable_sampler': True, 'g
  radient_accumulation_kwargs': None},
  adafactor=False,
  adam_beta1=0.9,
  adam_beta2=0.999,
  adam_epsilon=1e-08,
  auto_find_batch_size=False,
  bf16=False,
  bf16_full_eval=False,
  data_seed=None,
  dataloader_drop_last=False,
  dataloader_num_workers=0,
  dataloader_persistent_workers=False,
  dataloader_pin_memory=True,
  dataloader_prefetch_factor=None,

```

Congelar capas

Para realizar una afinación por capas en la que algunas capas se mantienen congeladas durante el entrenamiento ("congelación de capas"), se puede utilizar la funcionalidad de PyTorch para congelar los parámetros de las capas que no se quieren ajustar.

```
# Congelar las capas que no se quieren afinar
#for param in model.bert.parameters():
#    param.requires_grad = False
```

Si queremos congelar una capa intermedia específica en lugar de todas las capas BERT preentrenadas, se puede acceder a esa capa específica dentro del modelo y establecer su atributo `requires_grad` en `False`.

```
# Congelar una capa intermedia
#layer_to_freeze = 6 # Por ejemplo, congelar la sexta capa intermedia
#for param in model.bert.encoder.Layer[layer_to_freeze].parameters():
#    param.requires_grad = False
```

Agregar capas adicionales

La capa `qa_outputs` en el modelo BERT se encarga de generar las respuestas a partir de la representación de los tokens de la pregunta y del contexto.

Cuando agregamos capas adicionales al clasificador de salida del modelo BERT para preguntas y respuestas, estamos extendiendo la arquitectura del modelo al añadir más capas neuronales justo antes de las capas finales que predicen las respuestas. Estas capas adicionales se definen en `new_layers` y luego se agregan al final del clasificador de salida del modelo BERT utilizando el método `add_module()`.

Esto significa que las capas definidas en `new_layers` se integrarán en la arquitectura existente del modelo BERT para ajustarse mejor a la tarea específica de preguntas y respuestas.

Agregar capas adicionales puede permitir al modelo aprender representaciones más complejas y adaptarse mejor a los datos de entrenamiento, lo que puede resultar en un mejor rendimiento en la tarea de preguntas y respuestas.

```
# Agregar capas adicionales
num_added_layers = 2 # Definir el número de capas adicionales a agregar
hidden_size = model.config.hidden_size # Obtener el tamaño de la capa oculta del modelo BERT
new_layers = torch.nn.Sequential(
    torch.nn.Linear(hidden_size, hidden_size), # Capa lineal
    torch.nn.ReLU(), # Función de activación ReLU
    torch.nn.Dropout(0.1), # Capa de dropout para regularización
    torch.nn.Linear(hidden_size, hidden_size), # Segunda capa lineal
    torch.nn.ReLU(), # Segunda función de activación ReLU
    torch.nn.Dropout(0.1) # Segunda capa de dropout
)

# Agregar las capas adicionales al clasificador de salida del modelo BERT
model.qa_outputs.add_module('additional_layers', new_layers)
```

Observamos como la estructura del modelo BERT se ha ampliado con las nuevas capas

```
model
```

```
BertForQuestionAnswering(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(31002, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
```


Adam (Adaptive Moment Estimation)

"*Adaptive Moment Estimation*". Adam es un algoritmo de optimización ampliamente utilizado en el entrenamiento de redes neuronales, especialmente en problemas de aprendizaje profundo.

El algoritmo Adam combina las ideas de dos enfoques anteriores: el método de momentum, que mantiene un promedio ponderado de las actualizaciones de los pesos anteriores para acelerar la convergencia, y el método de la tasa de aprendizaje adaptativa, que adapta la tasa de aprendizaje de cada peso individualmente según su historia de gradientes.

Las principales características del algoritmo Adam son:

1. **Adaptabilidad de la tasa de aprendizaje:** Adam calcula una tasa de aprendizaje adaptativa para cada parámetro basándose en estimaciones de primer y segundo orden de los momentos del gradiente.
2. **Momentum:** Adam utiliza un término de momentum similar al utilizado en el método SGD con momento. Este término de momentum ayuda a suavizar el proceso de optimización y a acelerar la convergencia. El valor por defecto para el momentum en el optimizador Adam es 0.9. Esto significa que el algoritmo utiliza un coeficiente de 0.9 para acumular inercia en la dirección de los gradientes durante el proceso de optimización.
3. **Regularización:** Adam incorpora regularización L2 (*Weight Decay*) de forma implícita, lo que ayuda a prevenir el sobreajuste.

En resumen, Adam es un algoritmo de optimización muy efectivo y popular que combina la adaptabilidad de la tasa de aprendizaje con el momentum para mejorar la eficiencia y la estabilidad del entrenamiento de redes neuronales.

La regularización L2, también conocida como Weight Decay

Es una técnica utilizada en el entrenamiento de modelos de aprendizaje automático para prevenir el sobreajuste. Aquí están los detalles:

1. La regularización L2 agrega un término de penalización a la función de pérdida durante el entrenamiento del modelo.
2. Este término penaliza los pesos del modelo que son demasiado grandes, lo que ayuda a evitar que los coeficientes se vuelvan excesivamente sensibles a los datos de entrenamiento.
3. Durante el proceso de optimización, la función de pérdida se modifica para incluir un término adicional basado en la norma L2 de los pesos.
4. La norma L2 de un vector de pesos se calcula como la suma de los cuadrados de los valores de los pesos.
5. El término de penalización L2 se agrega a la función de pérdida original, multiplicado por un hiperparámetro llamado factor de regularización.
6. El objetivo es minimizar la función de pérdida total, que ahora incluye tanto la pérdida original como la penalización L2.
7. La regularización L2 ayuda a prevenir el sobreajuste al limitar la magnitud de los pesos del modelo.
8. Al penalizar los pesos grandes, se fomenta la simplicidad del modelo y se evita que se ajuste demasiado a los datos de entrenamiento ruidosos.

En resumen, la **regularización L2** (*Weight Decay*) es una herramienta útil para mejorar la generalización del modelo y garantizar que no se especialice demasiado en los datos de entrenamiento.

Detalles del optimizador Adam

1. **amsgrad**: Este parámetro determina si se utiliza la variante AMSGrad de Adam. AMSGrad modifica la regla de actualización para el promedio móvil de los gradientes al cuadrado.
2. **betas**: Estos son los coeficientes para las medias móviles exponenciales del gradiente y su cuadrado. Los valores predeterminados son (0.9, 0.999).
3. **eps**: Es una constante pequeña que se suma al denominador para evitar la división por cero.
4. **initial_lr**: La tasa de aprendizaje inicial.
5. **lr**: La tasa de aprendizaje actual.
6. **weight_decay**: La fuerza de regularización L2 (también conocida como decaimiento de peso). Controla la penalización sobre los pesos del modelo para prevenir el sobreajuste.

El scheduler LoRA con el optimizador SGD (Stochastic Gradient Descent)

Es posible aplicar el scheduler LoRA con el optimizador SGD (*Stochastic Gradient Descent*) para abordar el problema de un conjunto de datos pequeño. SGD es un algoritmo de optimización ampliamente utilizado en el aprendizaje profundo y es particularmente útil cuando se trabaja con conjuntos de datos pequeños, ya que puede ser más efectivo para explorar el espacio de búsqueda de manera más exhaustiva.

En nuestro caso, se utiliza el optimizador SGD con un momentum de 0.95 y una tasa de aprendizaje inicial de 0.01 ($lr = 1e-5$). Luego, se crea el scheduler LoRA utilizando la función LambdaLR de PyTorch, que ajusta la tasa de aprendizaje en función del número total de pasos de entrenamiento. Durante el entrenamiento, la tasa de aprendizaje se actualiza en cada iteración según el scheduler.

Se debe tener en cuenta que los valores de la tasa de aprendizaje inicial, el momentum y otros hiperparámetros pueden variar según el conjunto de datos y el modelo específico, por lo que es importante ajustarlos según sea necesario.

el ruido en los datos. Básicamente, el momentum acumula una fracción del gradiente pasado para acelerar el descenso del gradiente en la dirección dominante y disminuir las oscilaciones en direcciones menos importantes.

Si se aplica la función pero no se obtienen resultados satisfactorios, aquí hay algunas opciones que a considerar:

1. Tasa de aprendizaje: Probar con diferentes tasas de aprendizaje. A veces, una tasa de aprendizaje más baja puede ayudar a la convergencia, especialmente en conjuntos de datos pequeños.
2. Número de épocas: Aumentar el número de épocas de entrenamiento si el modelo aún no ha convergido. Esto puede ayudar a que el modelo aprenda patrones más complejos en los datos.
3. Regularización: Ajustar el parámetro de regularización (`weight_decay` en Adam, o `weight_decay` en SGD) para controlar el sobreajuste y mejorar el rendimiento del modelo en los datos de evaluación.
4. Explorar otros optimizadores: Además de SGD y Adam, hay otros optimizadores como RMSprop, Adagrad, etc. Podemos probar otros optimizadores para ver si mejoran el rendimiento del modelo.
5. Ajustar el momentum: El valor de momentum puede influir en la convergencia del modelo. Experimentar con diferentes valores para el momentum y observar cómo afecta el rendimiento del modelo.

El parámetro `momentum` en el optimizador SGD controla la velocidad y dirección de las actualizaciones de los pesos durante el entrenamiento. Es una medida de la inercia del proceso de optimización, es decir, cuánto se basa la dirección de la actualización en las actualizaciones anteriores.

Para un conjunto de datos de alrededor de 500 preguntas y respuestas, el valor óptimo para el parámetro momentum puede variar dependiendo de la naturaleza específica de los datos y del modelo. Sin embargo, típicamente se suele utilizar un valor entre 0.9 y 0.99. Estos valores son comunes en la práctica y suelen funcionar bien en muchas situaciones.

Aquí hay algunas pautas generales:

1. Valor Mínimo: El valor mínimo que se suele utilizar es 0, ya que significa que no hay inercia en las actualizaciones de los pesos y cada actualización se basa únicamente en el gradiente actual.
2. Valor Máximo: El valor máximo suele ser 1. Valores más altos pueden llevar a una mayor inercia, lo que puede ayudar a evitar que el optimizador se atasque en óptimos locales poco profundos.

Dicho esto, es recomendable realizar pruebas con diferentes valores de momentum en un rango de 0.1 a 0.99 para determinar cuál funciona mejor para tu conjunto de datos y modelo específicos. Vamos a comenzar con un valor de 0.9 y ajustarlo según los resultados de la validación. Después de varios entrenamientos, lo fijamos en 0.95.

Detalles del optimizador SGD

1. `dampening`: Este parámetro no se utiliza en la versión estándar de SGD. Es relevante en otros optimizadores, pero en SGD se mantiene en 0.
2. `differentiable`: Si está configurado en `False`, significa que el optimizador no es diferenciable. Esto podría ser útil en casos específicos donde se requiere un enfoque no diferenciable.
3. `initial_lr`: La tasa de aprendizaje inicial. Es el valor que se utiliza al principio del entrenamiento antes de que se realicen ajustes.
4. `lr`: La tasa de aprendizaje actual. Durante el entrenamiento, esta tasa se ajusta según el rendimiento del modelo.
5. `maximize`: Si está configurado en `False`, el optimizador minimiza la función de pérdida. Si se establece en `True`, maximiza una función objetivo (por ejemplo, en problemas de maximización).
6. `momentum`: El valor de momentum utilizado en SGD. El momentum es una técnica que acelera la convergencia al acumular inercia en la dirección de los gradientes.
7. `nesterov`: Si está configurado en `False`, se utiliza el método clásico de Nesterov para calcular el momentum. Si se establece en `True`, se utiliza la variante de Nesterov.
8. `weight_decay`: La fuerza de regularización L2 (Weight Decay). Controla la penalización sobre los pesos del modelo para prevenir el sobreajuste.

3. Finalmente, se pasa el scheduler junto con el optimizador al Trainer al crearlo.

Inicializa el Trainer con los argumentos y el scheduler

```
trainer = Trainer(  
    model=model,  
    args=training_args,  
    train_dataset=tokenized_datasets["train"],  
    eval_dataset=tokenized_datasets["validation"],  
    tokenizer=tokenizer,  
    data_collator=data_collator,  
    optimizers=(optimizer, scheduler)
```

DefaultDataCollator

```
from transformers import DefaultDataCollator  
data_collator = DefaultDataCollator()
```

DefaultDataCollator

El `DefaultDataCollator` es una parte esencial del proceso de entrenamiento de modelos de lenguaje natural (NLP). Es un recopilador de datos simple incluido en la biblioteca `Transformers`. Realiza un procesamiento básico en varios formatos de datos utilizados comúnmente para entrenar modelos de transformadores. Su función principal es preparar los datos de entrada de manera adecuada antes de pasarlos al modelo para el entrenamiento o la evaluación. Algunas de las tareas típicas que realiza un `data collator` incluyen:

1. Formato de datos:

Asume que tus datos están en forma de diccionarios, donde las claves representan tipos de datos (por ejemplo, "input_ids", "labels") y los valores son tensores que contienen los datos correspondientes.

2. Manejo especial de etiquetas:

Realiza un manejo específico para dos claves potenciales en los diccionarios:

- `label`: Si está presente, asume que esta clave contiene un solo valor (entero o flotante) que representa la etiqueta de la muestra.
- `label_ids`: Si está presente, asume que esta clave contiene una lista de valores que representan las etiquetas de la muestra (utilizada en tareas como la clasificación de secuencias).

`label`: Si está presente, asume que esta clave contiene un solo valor (entero o flotante) que representa la etiqueta de la muestra. `label_ids`: Si está presente, asume que esta clave contiene una lista de valores que representan las etiquetas de la muestra (utilizada en tareas como la clasificación de secuencias).

3. Relleno de tokens: Asegura que todas las secuencias de entrada tengan la misma longitud agregando tokens de relleno (generalmente tokens especiales como [PAD]) según sea necesario para igualar la longitud máxima en un lote.
4. Agrupación de ejemplos en lotes: Toma una lista de ejemplos individuales y los organiza en lotes de tamaño uniforme para el procesamiento eficiente en paralelo durante el entrenamiento.

El `DefaultDataCollator` proporciona una implementación predeterminada que maneja estas tareas comunes de manera eficiente. Sin embargo, en casos más específicos o personalizados, es posible que desees crear tu propio `data collator` para adaptarse a las necesidades de tu aplicación.

En resumen:

Se importa y crea una instancia de la clase `DefaultDataCollator`, un recopilador de datos básico adecuado para escenarios de entrenamiento simples con datos formateados en diccionarios en `Transformers`.

Inicialización de Trainer con los argumentos y optimizadores y Entrenamiento

El objeto `Trainer` es inicializado con varios parámetros, incluyendo el modelo preentrenado, los argumentos de entrenamiento, los conjuntos de datos tokenizados para entrenamiento y evaluación, el tokenizer, el `data_collator` (para el relleno dinámico de secuencias), y los optimizadores (que incluyen el optimizador y el scheduler).

El scheduler que está definido, se pasará como parte de la tupla de optimizadores al objeto `Trainer`, lo que significa que se utilizará durante el entrenamiento para ajustar dinámicamente la tasa de aprendizaje. En cada iteración de entrenamiento, el scheduler actualizará la tasa de aprendizaje según el esquema especificado (en este caso, LoRA).

Entonces, cuando se llama al método `trainer.train()`, el entrenamiento comienza utilizando el modelo, los datos de entrenamiento y evaluación, y los optimizadores con el scheduler incluido. Durante el entrenamiento, el scheduler se encargará de actualizar la tasa de aprendizaje según el número total de pasos de entrenamiento, siguiendo el esquema de LoRA que se ha definido. Esto permite una adaptación dinámica de la tasa de aprendizaje durante el proceso de entrenamiento, lo que puede ser beneficioso para obtener un mejor rendimiento, especialmente cuando se trabaja con conjuntos de datos pequeños.

Padding en la función de tokenización y el `DefaultDataCollator` en el `Trainer`.

Cuando se crea el Trainer para entrenar el modelo BERT con fine-tuning, puedes proporcionar un DataCollator como el DefaultDataCollator que se encargará de manejar el procesamiento final de los datos antes de pasarlos al modelo durante el entrenamiento. La función principal del DataCollator es realizar el relleno de tokens y la agrupación de ejemplos en lotes. En este caso, se ha utilizado:

- **padding="max_length"** en la tokenización, el DefaultDataCollator se encargará de agregar tokens de padding según sea necesario para igualar la longitud máxima en un lote.
- El **padding** (padding="max_length") especificado durante la tokenización se reflejará en el proceso de relleno de tokens realizado por el DefaultDataCollator para asegurar que todas las secuencias en un lote tengan la misma longitud. Esto es esencial para el entrenamiento eficiente del modelo.

```
# Inicializa el Trainer con los argumentos y el scheduler
trainer = Trainer(
    model=model, # Modelo preentrenado que deseas afinar
    args=training_args, # Argumentos de entrenamiento que has definido
    train_dataset=tokenized_datasets["train"], # Conjunto de datos de entrenamiento tokenizado
    eval_dataset=tokenized_datasets["validation"], # Conjunto de datos de evaluación tokenizado
    tokenizer=tokenizer, # Tokenizador
    data_collator=data_collator, # Padding: relleno de secuencia dinámica
)
```

Durante la ejecución del entrenador (trainer), las tres columnas representan la siguiente información:

1. **Epoch**: Esta columna indica el número de época actual durante el entrenamiento del modelo. Una época se refiere a una pasada completa a través de todo el conjunto de datos de entrenamiento.
2. **Training Loss** (Pérdida de entrenamiento): Esta columna muestra la pérdida calculada durante el entrenamiento en cada época. La pérdida de entrenamiento es una medida de qué tan bien se está ajustando el modelo a los datos de entrenamiento. Se calcula comparando las predicciones del modelo con las etiquetas verdaderas del conjunto de datos de entrenamiento.
3. **Validation Loss** (Pérdida de validación): Esta columna muestra la pérdida calculada durante la evaluación en el conjunto de datos de validación en cada época. La pérdida de validación es una medida de qué tan bien generaliza el modelo a datos que no ha visto durante el entrenamiento. Se calcula de manera similar a la pérdida de entrenamiento, pero utilizando el conjunto de datos de validación en lugar del conjunto de datos de entrenamiento.

El objetivo principal del entrenamiento es minimizar tanto la pérdida de entrenamiento como la de validación. Una **pérdida de entrenamiento baja** indica que el modelo está ajustando bien los datos de entrenamiento, mientras que una **pérdida de validación baja** indica que el modelo generaliza bien a datos no vistos. Sin embargo, es importante vigilar ambas pérdidas para detectar signos de sobreajuste (cuando la pérdida de entrenamiento sigue disminuyendo pero la pérdida de validación comienza a aumentar).

```
# Entrenar el modelo
%%time
trainer.train()
```

[510/510 07:54, Epoch 10/10]

Epoch	Training Loss	Validation Loss
1	No log	1.174107
2	1.387200	0.723761
3	1.387200	0.637797
4	0.271700	0.696578
5	0.271700	0.718242
6	0.091100	0.698468
7	0.091100	0.731071
8	0.032600	0.807296
9	0.032600	0.770403
10	0.017000	0.766648

CPU times: user 6min 54s, sys: 22.1 s, total: 7min 16s
Wall time: 7min 57s

```
TrainOutput(global_step=510, training_loss=0.35299577696942813, metrics={'train_runtime': 476.6901, 'train_samples_per_second': 8.475, 'train_steps_per_second': 1.07, 'total_flos': 1070298430095360.0, 'train_loss': 0.35299577696942813, 'epoch': 10.0})
```

Un valor de pérdida de entrenamiento de aproximadamente 0.5

Un valor de pérdida de entrenamiento de 0.5, por ejemplo, es relativamente bajo, lo que indica que el modelo está haciendo un buen trabajo en la tarea de entrenamiento. Sin embargo, la interpretación de la pérdida de entrenamiento depende en gran medida del contexto específico del problema y del rango típico de valores de pérdida para el problema en particular.

Aquí hay algunas consideraciones para evaluar si este valor de pérdida es bueno o no:

1. Comparación con la pérdida de validación: Es importante comparar la pérdida de entrenamiento con la pérdida de validación. Si la pérdida de entrenamiento es baja pero la pérdida de validación es alta, podría indicar que el modelo está sobreajustando los datos de entrenamiento y no generaliza bien a datos nuevos.
2. Comparación con iteraciones anteriores: Si tienes registros de pérdida de entrenamiento de iteraciones anteriores, puedes comparar el valor actual con los valores anteriores para ver si ha habido una mejora o empeoramiento.
3. Contexto del problema: Algunos problemas pueden tener pérdidas de entrenamiento más bajas que otros debido a la complejidad de la tarea. Es importante considerar el contexto del problema y si este valor de pérdida es razonable dadas las circunstancias.
4. Objetivo del entrenamiento: Dependiendo de tus objetivos de entrenamiento, un valor de pérdida de entrenamiento determinado puede ser considerado como aceptable o no. Si el objetivo es simplemente entrenar un modelo para una tarea específica, un valor de pérdida de entrenamiento de 0.5 puede ser suficiente. Sin embargo, si estás buscando un rendimiento óptimo en una métrica específica, es posible que necesites optimizar aún más el modelo.

En resumen, un valor de pérdida de entrenamiento de 0.5 parece ser bastante bajo, pero es importante considerar otros factores, como la pérdida de validación y el contexto del problema, para evaluar si este valor es satisfactorio para las necesidades específicas.

Método `train()`

El método `train()` se encargará de iterar sobre el conjunto de datos de entrenamiento y ajustar el modelo a los datos.

Análisis del resultado de `trainer.train()`:

El resultado que obtienes al ejecutar `import torch, trainer.train()` en PyTorch representa el rendimiento de una época de entrenamiento para tu modelo. Veamos qué significa cada elemento:

1. `TrainOutput`: Esta es una estructura de datos personalizada que contiene información sobre el entrenamiento.
2. `global_step ()`: Indica el número de iteración o paso global actual en el proceso de entrenamiento. Comienza en 0 y aumenta después de procesar cada lote.
3. `training_loss ()`: Este es el valor promedio de la pérdida calculado en todos los lotes de la época actual. Un valor de pérdida más bajo generalmente indica un mejor rendimiento del modelo.
4. `metrics (diccionario)`: Este diccionario contiene varias métricas registradas durante el entrenamiento:
 - `train_runtime ()`: Tiempo que tardó en entrenar el modelo durante toda la época (en segundos).
 - `train_samples_per_second ()`: Promedio de muestras de entrenamiento procesadas por segundo durante la época.
 - `train_steps_per_second ()`: Promedio de pasos de entrenamiento (lotes) procesados por segundo durante la época.
 - `total_flos ()`: Número total de FLOPs (operaciones de punto flotante) realizadas durante el entrenamiento en la época. Esta métrica puede ser útil para comprender el uso computacional.
 - `train_loss ()`: Duplicado del valor de la pérdida de entrenamiento mencionado anteriormente.
 - `epoch ()`: Especifica el número de época actual.

En resumen, este resultado proporciona información valiosa sobre el progreso del entrenamiento de tu modelo después de completar todas las épocas. Se puede usar esta información para:

- Monitorizar la pérdida: Rastrear cómo la pérdida de entrenamiento disminuye con las épocas, lo que indica una posible convergencia.
- Evaluar la velocidad de entrenamiento: Analizar `train_runtime`, `train_samples_per_second` y `train_steps_per_second` para comprender la eficiencia del entrenamiento.
- Optimizar el uso de recursos: Evaluar `total_flos` para comprender los requisitos computacionales del entrenamiento.

Se recomienda monitorizar estas métricas a lo largo de varias épocas para obtener una imagen más clara del proceso de entrenamiento.

Cuando durante el entrenamiento de un modelo se muestra "No log" en el "Training Loss", significa que no se ha registrado ningún valor para la pérdida durante ese paso de entrenamiento específico. Esto podría ocurrir por varias razones:

1. **Problemas durante el entrenamiento:** Puede haber habido un error o un problema durante el proceso de entrenamiento que impidió que se registrara la pérdida.
2. **Configuración incorrecta:** La configuración del entrenamiento o la forma en que se está calculando la pérdida podrían ser incorrectas, lo que podría resultar en valores inesperados o la falta de registro de la pérdida.
3. **Falta de retroalimentación:** Es posible que no se esté proporcionando la retroalimentación adecuada al modelo durante el entrenamiento, lo que podría hacer que no se calcule o registre la pérdida.

En resumen, "No log" en el "Training Loss" indica que no se ha registrado ningún valor de pérdida durante ese paso de entrenamiento, lo cual es anormal y requiere una investigación adicional para comprender y corregir el problema.

Aumentar la retroalimentación

Para aumentar la retroalimentación y asegurar de que se registra algún valor de pérdida durante el proceso de entrenamiento, se pueden considerar las siguientes opciones:

1. **Verificar la configuración del optimizador:** Asegúrese de que se está utilizando un optimizador adecuado para la tarea de entrenamiento. Experimentar con diferentes optimizadores y ajustar los hiperparámetros correspondientes, como la tasa de aprendizaje y el decaimiento de peso, para mejorar la convergencia del modelo.
2. **Revisar la función de pérdida:** Verificar que se está utilizando la función de pérdida correcta para tu tarea específica. Dependiendo del tipo de problema que se aborda (clasificación, regresión, etc.), es importante elegir una función de pérdida adecuada que refleje la naturaleza de tus datos y tus objetivos de entrenamiento.
3. **Aumentar el tamaño del lote:** A veces, el uso de lotes más grandes durante el entrenamiento puede proporcionar una retroalimentación más estable y significativa para el modelo, lo que puede ayudar a que se registren valores de pérdida durante el proceso de entrenamiento. Para aumentar el tamaño del lote, simplemente se puede incrementar el valor de `batch_size` en los argumentos de entrenamiento. Por ejemplo, si se desea duplicar el tamaño del lote, se podría establecer `batch_size` en el doble del valor actual. Sin embargo, hay que tener en cuenta que aumentar demasiado el tamaño del lote puede consumir más memoria y recursos computacionales. Es importante encontrar un equilibrio entre un tamaño de lote más grande y los recursos disponibles.
4. **Ajustar la arquitectura del modelo:** Si utilizamos una arquitectura de red neuronal compleja, hay que considerar simplificarla o ajustarla para reducir la posibilidad de problemas de convergencia durante el entrenamiento.
5. **Verificar la calidad de los datos de entrada:** Asegurar de que los datos de entrenamiento estén limpios y bien preparados. Los datos de baja calidad o mal formateados pueden dificultar el entrenamiento del modelo y dificultar la convergencia.
6. **Agregar más datos de entrenamiento:** Si es posible, aumentar el tamaño de tu conjunto de datos de entrenamiento. Más datos de entrenamiento pueden proporcionar una retroalimentación más rica y variada para el modelo, lo que puede ayudar a mejorar la calidad del entrenamiento y la convergencia del modelo.
7. **Explorar técnicas de regularización:** Considerar el uso de técnicas de regularización, como la deserción (*dropout*) o la penalización L2, para ayudar a controlar el sobreajuste y mejorar la estabilidad del entrenamiento.

Al experimentar con estas opciones y realizar ajustes en el proceso de entrenamiento, se debería poder aumentar la retroalimentación y visualizar de que se registren valores de pérdida durante el entrenamiento del modelo.

Optimizador

Dado que estamos trabajando con un conjunto de datos relativamente pequeño (500), hay algunas modificaciones que se pueden considerar en el código del optimizador para mejorar el entrenamiento de tu modelo. Algunas opciones:

1. **Tasa de aprendizaje inicial (`learning_rate`):** Dado que tenemos un conjunto de datos pequeño, es posible ajustar la tasa de aprendizaje inicial para controlar la velocidad a la que el modelo aprende de los datos. Una tasa de aprendizaje más baja puede ayudar a evitar un sobreajuste rápido en conjuntos de datos pequeños.
2. **Decaimiento de peso (`weight_decay`):** El decaimiento de peso se utiliza para penalizar los pesos grandes en la función de pérdida durante el entrenamiento, lo que puede ayudar a prevenir el sobreajuste. Sin embargo, en conjuntos de datos pequeños, un decaimiento de peso demasiado alto podría limitar la capacidad del modelo para aprender de los datos. Hay que considerar reducir el valor de decaimiento de peso o incluso omitirlo inicialmente para ver cómo afecta al entrenamiento.
3. **Explorar otras optimizaciones:** Además de **Adam**, se pueden explorar otras opciones de optimización que podrían funcionar mejor para conjuntos de datos pequeños. Por ejemplo, **SGD (Descenso de Gradiente Estocástico)** con impulso (`momentum`) o adaptación del tamaño del paso (`learning_rate scheduling`) podría ser más efectivo en este escenario.

Es importante realizar experimentos y ajustes iterativos en la configuración del optimizador para encontrar lo que funciona mejor para tu conjunto de datos y tu modelo específico.

Guardar el modelo entrenado en disco

La instrucción `trainer.save_model()` se utiliza en el contexto del paquete `transformers` de Hugging Face en Python y sirve para guardar el modelo entrenado en disco. Esta función es parte del objeto `Trainer` que se utiliza para entrenar modelos de `transformers`.

Cuando entrenas un modelo con el `Trainer`, después de que el entrenamiento ha concluido y estás satisfecho con el rendimiento del modelo, puedes llamar a `trainer.save_model()` para guardar todos los pesos y configuraciones del modelo en un directorio específico.

Cuando utilizas `trainer.save_model()`, el modelo entrenado se guarda en un directorio específico en tu sistema de archivos. Por defecto, este directorio se llama `output` y se crea en el directorio de trabajo actual. Dentro de este directorio `output`, se creará una carpeta adicional con el nombre del modelo, y dentro de esta carpeta se guardarán los archivos relacionados con el modelo, como los pesos del modelo y los archivos de configuración.

Si deseas especificar una ubicación diferente para guardar el modelo, puedes pasar la ruta de destino como argumento a la función `trainer.save_model()`:

En caso de no existir la carpeta `MODELOS`, se crea automáticamente y los 7 archivos del modelo se guardan.

```
trainer.save_model('/content/drive/MyDrive/MODELOS/OptimizadorAdamW')
```

Se crean estos archivos:

1. `config.json`
2. `model.safetensors`
3. `special_tokens_map.json`
4. `tokenizer.json`
5. `tokenizer_config.json`
6. `training_args.bin`
7. `vocab.txt`

Explicación de cada archivo guardado por `trainer.save_model()`:

1. `config.json`: Este archivo contiene la configuración del modelo, incluyendo los hiperparámetros, la arquitectura y cualquier otra configuración necesaria para recrear el modelo.
2. `model.safetensors`: Este archivo contiene los pesos del modelo entrenado. Es esencial para cargar el modelo entrenado en el futuro y realizar inferencias o continuar el entrenamiento desde donde lo dejaste.
3. `special_tokens_map.json`: Este archivo mapea los tokens especiales utilizados por el tokenizador a sus identificadores correspondientes. Estos tokens especiales pueden ser tokens de inicio, tokens de fin, tokens de relleno, etc.
4. `tokenizer.json`: Este archivo contiene la información necesaria para recrear el tokenizador utilizado durante el entrenamiento del modelo. Incluye el vocabulario, las configuraciones de tokenización y cualquier otra información relevante.
5. `tokenizer_config.json`: Similar al archivo `tokenizer.json`, este archivo contiene la configuración del tokenizador.
6. `training_args.bin`: Este archivo almacena los argumentos de entrenamiento utilizados durante el entrenamiento del modelo. Esto incluye información sobre la configuración del entrenamiento, como la tasa de aprendizaje, el tamaño del lote, el número de épocas, etc.
7. `vocab.txt`: Este archivo contiene el vocabulario utilizado por el tokenizador. Esencialmente, mapea los tokens a sus identificadores correspondientes y viceversa. Este archivo se utiliza durante la tokenización del texto de entrada durante la inferencia o el entrenamiento.

```
trainer.save_model('/content/drive/MyDrive/MODELOS/OptimizadorAdamW')
```

Ver la carpeta `content` (`/content/drive/MyDrive/MODELOS`)

Análisis y conclusiones sobre la calidad del modelo:

Basándonos en los resultados obtenidos en el entrenamiento:

- Pérdida de entrenamiento (`loss`): La pérdida de entrenamiento parece ser relativamente baja, lo que indica un buen rendimiento del modelo en esta etapa inicial.
- Velocidad de entrenamiento: Los valores de `train_samples_per_second` y `train_steps_per_second` sugieren una velocidad de entrenamiento razonable.

Aspectos a considerar:

- Número de épocas de entrenamiento: Variar diferentes número de épocas.
- Conjunto de datos: Tamaño.
- Tarea: Tipo de tarea (generación de texto).
- Métricas de evaluación: Métricas relevantes para la tarea específica (precisión, F1 score, etc.).

Siguientes procesos:

- Continuar el entrenamiento: Es recomendable continuar el entrenamiento durante más épocas para observar cómo evoluciona la pérdida y otras métricas relevantes.
- Evaluar el rendimiento: Una vez finalizado el entrenamiento, se debe evaluar el rendimiento del modelo en el conjunto de datos de validación independiente para obtener una medida precisa de su calidad.
- Comparar con otros modelos: Si es posible, comparar el rendimiento del modelo con otros modelos existentes para la misma tarea.

En resumen:

- Los resultados iniciales parecen indicar un modelo con un buen potencial. Sin embargo, se necesita más información y evaluación para determinar la calidad final del modelo en el contexto de la tarea específica.
- La calidad de un modelo depende en gran medida de la calidad del conjunto de datos, la tarea a realizar y las métricas de evaluación utilizadas.
- Es importante comparar el modelo con otros modelos existentes para tener una referencia.
- El entrenamiento y la evaluación de modelos pueden ser procesos iterativos que requieren ajustes y refinamientos.

Prueba con una instancia del conjunto de entrenamiento

Usaremos la instancia con índice 7 en el conjunto de datos como ejemplo.

```
instance = data['train'][12]
context = instance['context']
question = instance['question']
```

context

'Sí, hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución, bien por obras de mejora o por situaciones sobrevenidas, y la duración estimada de la suspensión temporal del suministro. Este servicio se ha desarrollado para que puedas adoptar las medidas necesarias para reducir el impacto que estas intervenciones pudieran ocasionar en tu actividad diaria. Ponte en contacto con nosotros a través del teléfono gratuito 900 365 365, o date de alta desde tu perfil (para ello deberás estar registrado).'

question

'¿Hay alguna app para saber si no va a haber agua?'

Encuentrar la respuesta dada y su posición inicial en el contexto:

```
instance['answers']
```

```
{'answer_start': [4],
 'text': ['Hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución']}
```

```
given_answer = instance['answers']['text'][0] # Suponiendo que la primera respuesta sea la correcta
given_answer_start = instance['answers']['answer_start'][0]
given_answer, given_answer_start
```

```
('Hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución',
 4)
```

Tokenizar los datos de un ejemplo con el modelo

```
# Tokenizar Los datos con el tokenizer1
inputs = tokenizer(question, context, return_tensors='pt', max_length=512, truncation=True)
```

Aplicar el modelo BERT a los datos de un ejemplo

torch es una biblioteca de Python

`torch` es una biblioteca de Python que se utiliza principalmente para cálculos numéricos y operaciones con tensores. Breve explicación:

1. Tensores:

- En el contexto de aprendizaje profundo (deep learning), los tensores son estructuras de datos similares a matrices multidimensionales.
- PyTorch proporciona una implementación eficiente de tensores, lo que permite realizar cálculos en CPU o GPU.
- Los tensores son fundamentales para representar datos, parámetros de modelos y gradientes durante el entrenamiento.

2. Funcionalidades de PyTorch:

- Autograd: PyTorch utiliza un sistema de seguimiento automático de gradientes llamado autograd. Esto permite calcular automáticamente gradientes para funciones definidas por el usuario.
- Redes neuronales: PyTorch facilita la creación y entrenamiento de redes neuronales. Puedes definir tus propias arquitecturas de red o utilizar modelos preentrenados.
- Optimización: PyTorch proporciona optimizadores como SGD, Adam, etc., para ajustar los parámetros de los modelos durante el entrenamiento.
- Manipulación de datos: Puedes cargar datos, transformarlos y prepararlos para el entrenamiento utilizando las herramientas de manipulación de datos de PyTorch.

3. Compatibilidad con GPU:

- PyTorch está diseñado para aprovechar la potencia de las GPUs.
- Al mover tensores y modelos a la GPU (usando `to(device)`), puedes acelerar significativamente los cálculos.

En resumen, `torch` te permite acceder a todas estas funcionalidades y herramientas para trabajar con tensores, construir modelos de aprendizaje profundo y realizar cálculos numéricos. Es una biblioteca esencial en el campo del aprendizaje profundo.

```
import torch
```

Selección del Dispositivo (CPU o GPU):

Crea un objeto `device` que representa la unidad de procesamiento en la que se ejecutará el modelo.

Si hay una GPU disponible, se selecciona "cuda" (para aprovechar la aceleración de GPU). De lo contrario, se utiliza "cpu".

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Mover el Modelo al Dispositivo Seleccionado:

`model.to(device)` mueve el modelo `model` (un modelo preentrenado de BERT) al dispositivo especificado.

Esto es necesario para asegurarse de que los cálculos se realicen en la GPU (si está disponible) o en la CPU.

```
model.to(device)
```

```
BertForQuestionAnswering(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(31002, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,)), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
)
```


Obtención de la Respuesta Predicha:

`start_idx` y `end_idx` son los índices con las puntuaciones máximas para el inicio y el final de la respuesta, respectivamente.

Se extraen los tokens correspondientes a esa ventana de índices.

Luego, se convierten los tokens en una cadena de texto (respuesta predicha).

```
# Obtener la respuesta prevista
start_idx = torch.argmax(output.start_logits)
end_idx = torch.argmax(output.end_logits)

predicted_answer = tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(inputs['input_ids'])[0][start_idx:end_idx +
```

Información Adicional:

`start_idx.item()` y `end_idx.item()` devuelven los valores enteros de los índices.

```
predicted_answer, start_idx, end_idx, start_idx.item(), end_idx.item()
```

```
('hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución',
 tensor(17, device='cuda:0'),
 tensor(40, device='cuda:0'),
 17,
 40)
```

En resumen, este código se utiliza para aplicar un modelo preentrenado de BERT a una instancia de datos (pregunta y contexto), obtener las puntuaciones de inicio y fin para la respuesta y convertir esos índices en una respuesta predicha.

Evaluar el resultado de los datos de ejemplo con el modelo inicial

```
predicted_answer.lower()
```

```
'hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución'
```

```
start_idx
```

```
tensor(17, device='cuda:0')
```

```
end_idx
```

```
tensor(40, device='cuda:0')
```

```
given_answer.lower()
```

```
'hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución'
```

```
predicted_answer.lower()
```

```
'hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución'
```

```
given_answer.lower()
```

```
'hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución'
```

```
correct = (predicted_answer.lower() == given_answer.lower())
evaluation = 'Correcto' if correct else f'Incorrecto (Predicción: {predicted_answer}, Solución: {given_answer})'

print(evaluation)
```

Correcto

Observación: Si la evaluación es correcta, el accuracy será 1. Pero si la evaluación difiere en algún carácter, el accuracy será 0, pero es posible que la respuesta tenga validez. Por ejemplo:

1. Predicción: sí, hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución
2. Solución: Hemos puesto en marcha un nuevo servicio para comunicarte de manera inmediata las incidencias que pudieran producirse en la red de distribución

Entonces, en la posterior evaluación del dataset, basándonos en métricas como el accuracy, debemos tener en cuenta que el accuracy posiblemente reflejará un porcentaje inferior de aciertos. Pero es una métrica que nos vale para empezar a tomar decisiones.

Función para evaluar una sola instancia

La función para evaluar una sola instancia utilizando un modelo preentrenado de BERT. Vamos a desglosar cada paso:

1. Preparación de datos:

- La función recibe una instancia que contiene tres elementos clave:

- `context`: El contexto o el texto en el que se encuentra la pregunta.

- `question`: La pregunta que se desea responder.

- `given_answer`: La respuesta correcta proporcionada en la instancia (suponemos que es la primera respuesta).

2. Tokenización:

- Primero, se tokenizan los datos utilizando el tokenizador BERT.
- La función `tokenizer` toma la pregunta y el contexto, y devuelve una representación tokenizada de los mismos.
- Se especifica `return_tensors='pt'` para obtener tensores de PyTorch como salida.

3. Preparación de entradas para el modelo:

- Los datos tokenizados se convierten en tensores de PyTorch y se envían al dispositivo (CPU o GPU) especificado (variable `device`).
- Esto es necesario para que el modelo BERT pueda procesar los datos en el mismo dispositivo.

4. Aplicación del modelo BERT:

- Se utiliza el modelo preentrenado de BERT para procesar las entradas.
- La línea `output = model(**inputs)` realiza la inferencia en el modelo.
- La salida contiene dos tensores: `start_logits` y `end_logits`, que representan las puntuaciones de inicio y fin para la respuesta.

5. Obtención de la respuesta predicha:

- Se encuentra el índice con la puntuación máxima tanto para el inicio como para el final de la respuesta.
- Luego, se extraen los tokens correspondientes a esa ventana de índices.
- Finalmente, se convierten los tokens en una cadena de texto (respuesta predicha).

6. Comparación con la respuesta dada:

- La respuesta predicha se compara con la respuesta proporcionada en la instancia.
- Si son iguales (ignorando mayúsculas y minúsculas), la función devuelve `True`, lo que indica que la predicción fue correcta.

En resumen, esta función toma una instancia con contexto, pregunta y respuesta correcta, tokeniza los datos, aplica el modelo BERT y compara la respuesta predicha con la respuesta dada. Si coinciden, devuelve `True`. Es una parte fundamental en la evaluación de modelos de procesamiento del lenguaje natural como BERT.

```
# Función para evaluar una sola instancia
def evaluate_instance(instance, device):
    context = instance['context']
    question = instance['question']
    given_answer = instance['answers']['text'][0] # Suponiendo que La primera respuesta sea La correcta

    # Tokenizar Los datos
    inputs = tokenizer(question, context, return_tensors='pt', max_length=512, truncation=True)

    inputs = {k: v.to(device) for k, v in inputs.items()}

    # Aplicación del modelo BERT
    with torch.no_grad(): # No es necesario calcular gradientes
        output = model(**inputs)

    # Obtén La respuesta predicha (predicted answer)
    start_idx = torch.argmax(output.start_logits)
    end_idx = torch.argmax(output.end_logits)
    predicted_answer = tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(inputs['input_ids'][0][start_idx:end_idx]))

    return predicted_answer.lower() == given_answer.lower()
```

La librería `tqdm` en Python

La librería `tqdm` de Python es una librería muy útil que proporciona una barra de progreso visual y una estimación del tiempo restante para iteraciones en bucles.

Cuando utilizas `tqdm` en un bucle `for`, envuelves la iteración con la función `tqdm()` para mostrar una barra de progreso.

En nuestro caso, `tqdm(range(total_count))` envuelve la iteración en un bucle sobre un rango del número de registros que tiene el Dataset `train`, mostrando una barra de progreso en la consola que se actualiza a medida que avanza el bucle.

Además de la barra de progreso, `tqdm` también proporciona otras características útiles, como la capacidad de mostrar el tiempo transcurrido, el tiempo restante estimado y la tasa de iteración. Esto hace que sea fácil monitorear el progreso de los bucles, especialmente cuando tienes iteraciones largas o bucles anidados.

```
from tqdm import tqdm
```

Aplicar el modelo en un conjunto de instancias

```
# Evalúe el número de instancias
correct_count = 0
total_count = len(data['train'])
total_count
```

484

```
# Iteración en un bucle con tqdm
for i in tqdm(range(total_count)):
    correct_count += evaluate_instance(data['train'][i], device)
```

```
100%|██████████| 484/484 [00:05<00:00, 73.07it/s]
```

```
# Calcule y genere La precisión (accuracy)
accuracy = correct_count / total_count
print(f'Accuracy: {accuracy * 100:.2f}%')
```

Accuracy: 64.85%

```
# Evalúe el número de instancias
correct_count = 0
total_count = len(data['validation'])
total_count
```

100

```
# Iteración en un bucle con tqdm
for i in tqdm(range(total_count)):
    correct_count += evaluate_instance(data['validation'][i], device)
```

```
100%|██████████| 102/102 [00:01<00:00, 57.23it/s]
```

```
# Calcule y genere la precisión (accuracy)
accuracy = correct_count / total_count
print(f'Accuracy: {accuracy * 100:.2f}%')
```

Accuracy: 56.86%

```
def generate_answer(context, question, model, tokenizer, device):
    # Tokenizar Los datos
    inputs = tokenizer(question, context, return_tensors='pt', max_length=512, truncation=True)

    inputs = {k: v.to(device) for k, v in inputs.items()}

    # Aplicación del modelo BERT
    with torch.no_grad(): # No es necesario calcular gradientes
        output = model(**inputs)

    # Obtén la respuesta predicha (predicted answer)
    start_idx = torch.argmax(output.start_logits)
    end_idx = torch.argmax(output.end_logits)
    predicted_answer = tokenizer.convert_tokens_to_string(tokenizer.convert_ids_to_tokens(inputs['input_ids'][0][start_idx:end_idx]))

    return predicted_answer.lower()
```

```
# Ejemplo de uso:
context = "El Canal de Isabel II es una empresa pública española encargada de la gestión del ciclo integral del agua en la Comunidad de Madrid."
question = "¿Cuál es la misión de Canal de Isabel II?"
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

answer = generate_answer(context, question, model, tokenizer, device)
print("Respuesta generada por el asistente:", answer)
```

Respuesta generada por el asistente: su misión es principalmente el abastecimiento de la población

Cargar el modelo que guarde después del entrenamiento

Para cargar un modelo que ha sido guardado utilizando `trainer.save_model()`, puedes seguir estos pasos utilizando la biblioteca transformers de Hugging Face en Python:

```
# Ruta donde se guardó el modelo
ruta_modelo = '/content/drive/MyDrive/MODELOS/OptimizadorAdamW'
ruta_modelo
```

```
'/content/drive/MyDrive/MODELOS/OptimizadorAdamW'
```

```
# Cargar el modelo y el tokenizador
tokenizer2 = AutoTokenizer.from_pretrained(ruta_modelo)
tokenizer2
```

```
BertTokenizerFast(name_or_path='/content/drive/MyDrive/MODELOS/OptimizadorAdamW', vocab_size=31002, model_max_length=512, is_fast=True, padding_side='right', truncation_side='right', special_tokens={'unk_token': '[UNK]', 'sep_token': '[SEP]', 'pad_token': '[PAD]', 'cls_token': '[CLS]', 'mask_token': '[MASK]'}, clean_up_tokenization_spaces=True), added_tokens_decoder={
  0: AddedToken("[MASK]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  1: AddedToken("[PAD]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  3: AddedToken("[UNK]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  4: AddedToken("[CLS]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
  5: AddedToken("[SEP]", rstrip=False, lstrip=False, single_word=False, normalized=False, special=True),
}
```

```
# Cargar el modelo
model2 = AutoModelForQuestionAnswering.from_pretrained(ruta_modelo)
```

Some weights of the model checkpoint at /content/drive/MyDrive/Modelos/OptimizadorAdamW were not used when initializing BertForQuestionAnswering: ['qa_outputs.additional_layers.0.bias', 'qa_outputs.additional_layers.0.weight', 'qa_outputs.additional_layers.3.bias', 'qa_outputs.additional_layers.3.weight']

- This IS expected if you are initializing BertForQuestionAnswering from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForQuestionAnswering from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

Cuando guardas o cargas un modelo BERT con capas adicionales agregadas al clasificador de salida, la estructura del modelo no mostrará explícitamente esas capas adicionales en la lista de capas del modelo. Sin embargo, las capas adicionales estarán presentes en el modelo guardado y se cargarán junto con el resto del modelo al cargarlo de nuevo.

El modelo BERT preentrenado tiene una estructura fija que incluye capas específicas para la tarea preentrenada, como clasificación de texto o preguntas y respuestas. Cuando agregas capas adicionales al clasificador de salida, estas se integran en la arquitectura del modelo como módulos secuenciales o lineales, pero no se incluyen explícitamente en la estructura del modelo BERT preentrenado.

Cuando guardas el modelo usando `model.save_pretrained()` y luego lo cargas usando `AutoModelForQuestionAnswering.from_pretrained()`, las capas adicionales que se agregaron al clasificador de salida se guardarán y cargarán junto con el modelo BERT preentrenado. Sin embargo, al imprimir la estructura del modelo, solo se verán las capas preexistentes del modelo BERT original, aunque las capas adicionales estarán presentes y funcionando correctamente en el modelo cargado.

```
model2
```

```
BertForQuestionAnswering(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(31002, 768, padding_idx=1)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
  )
  (qa_outputs): Linear(in_features=768, out_features=2, bias=True)
)
```

Evaluar el resultado de los datos de ejemplo con el modelo 2 guardado y cargado

Para evaluar los resultados del asistente basado en un modelo preentrenado de BERT con fine-tuning en un conjunto de datos etiquetado, podemos utilizar varias métricas.

1. Exactitud (**Accuracy**): Esta métrica mide la proporción de respuestas correctas proporcionadas por el asistente en comparación con las respuestas etiquetadas en el conjunto de datos. La exactitud es útil para evaluar el rendimiento general del modelo en todas las preguntas.
2. Precisión (**Precision**) y Recuperación (**Recall**): Estas métricas son útiles para evaluar el rendimiento del asistente en preguntas específicas. La precisión mide la proporción de respuestas correctas entre todas las respuestas proporcionadas por el asistente para una pregunta en particular, mientras que la recuperación mide la proporción de respuestas correctas proporcionadas por el asistente en relación con todas las respuestas correctas posibles para esa pregunta en el conjunto de datos.
3. **F1-Score**: Esta métrica es la media armónica de precisión y recuperación y proporciona una medida única del rendimiento del asistente en una pregunta específica. Es útil cuando se desea equilibrar la precisión y la recuperación.

```
# Función para evaluar una sola instancia del modelo2
def evaluate_instance_model2(instance, device):
    context = instance['context']
    question = instance['question']
    given_answer = instance['answers']['text'][0] # Suponiendo que La primera respuesta sea La correcta

    # Tokenizar2 Los datos
    inputs = tokenizer2(question, context, return_tensors='pt', max_length=512, truncation=True)

    inputs = {k: v.to(device) for k, v in inputs.items()}

    # Aplicación del modelo BERT
    with torch.no_grad(): # No es necesario calcular gradientes
        output = model2(**inputs)

    # Obtén La respuesta predicha (predicted answer)
    start_idx = torch.argmax(output.start_logits)
    end_idx = torch.argmax(output.end_logits)
    predicted_answer = tokenizer2.convert_tokens_to_string(tokenizer2.convert_ids_to_tokens(inputs['input_ids'][0][start_idx:end_idx]))

    return predicted_answer.lower() == given_answer.lower()
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model2.to(device)
inputs = {k: v.to(device) for k, v in inputs.items()}
```

Evaluación de las respuestas sobre el conjunto de datos de train

```
def evaluate_instance2(instance, generated_response):
    # Obtén La pregunta y La respuesta de referencia de La instancia
    question = instance['question']
    reference_response = instance['answers']['text'][0] # Suponiendo que La primera respuesta sea La correcta

    # Calcula La precisión, el recall y el F1-Score
    precision = len(set(generated_response.split()) & set(reference_response.split())) / len(generated_response.split())
    recall = len(set(generated_response.split()) & set(reference_response.split())) / len(reference_response.split())
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

    # Calcula La precisión (accuracy)
    accuracy = 1 if generated_response.lower() == reference_response.lower() else 0

    return {
        'precision': precision,
        'recall': recall,
        'f1_score': f1_score,
        'accuracy': accuracy
    }
```

```

precision_total = 0
recall_total = 0
f1_score_total = 0
accuracy_total = 0

for i in tqdm(range(total_count)):
    instance = data['train'][i]
    generated_response = generate_answer(context, question, model2, tokenizer2, device)
    evaluation_result = evaluate_instance2(instance, generated_response)

    # Agrega los resultados de precisión, recall, F1-Score y accuracy a los totales
    precision_total += evaluation_result['precision']
    recall_total += evaluation_result['recall']
    f1_score_total += evaluation_result['f1_score']
    accuracy_total += evaluation_result['accuracy']

# Calcula las métricas promedio
precision_avg = precision_total / total_count
recall_avg = recall_total / total_count
f1_score_avg = f1_score_total / total_count
accuracy_avg = accuracy_total / total_count
  
```

```
100%|██████████| 404/404 [00:05<00:00, 70.18it/s]
```

```

# Imprime los resultados
print(f'Precision: {precision_avg}')
print(f'Recall: {recall_avg}')
print(f'F1-Score: {f1_score_avg}')
print(f'Accuracy: {accuracy_avg}')
  
```

```

Precision: 0.2021452145214528
Recall: 0.0865754383946521
F1-Score: 0.1143769991589111
Accuracy: 0.5686274509803921
  
```

Este código asegura que la precisión, recall y F1-Score se calculen correctamente sin importar la longitud de las respuestas generadas y de referencia. Las respuestas se convierten a minúsculas antes de calcular las métricas para evitar discrepancias debido a diferencias de capitalización.

1. **Precisión (Precision):** La precisión mide la fracción de instancias recuperadas que son relevantes. En este caso, se calcula la precisión dividiendo el tamaño de la intersección entre las palabras de la respuesta generada y las palabras de la respuesta de referencia entre el tamaño de la respuesta generada.
2. **Recall:** El recall mide la fracción de instancias relevantes que han sido recuperadas sobre el total de instancias relevantes. De manera similar a la precisión, estás calculando el recall dividiendo el tamaño de la intersección entre las palabras de la respuesta generada y las palabras de la respuesta de referencia entre el tamaño de la respuesta de referencia.
3. **F1-Score:** El F1-Score es la media armónica de precisión y recall y se utiliza para proporcionar un equilibrio entre precisión y recall. E
4. **Accuracy:** La precisión (accuracy) simplemente mide la fracción de instancias clasificadas correctamente sobre el total de instancias. En este caso, se compara la respuesta generada con la respuesta de referencia. Es importante destacar que el accuracy no tiene en cuenta la relación token a token entre la respuesta generada y la respuesta de referencia, lo que puede explicar por qué los resultados de precisión, recall y F1-Score no coinciden con el accuracy.

Función para calcular las métricas de precisión, recall y F1-Score

```

# Función para calcular las métricas de precisión, recall y F1-Score
def calculate_metrics(reference_response, generated_response):
    # Convertir las respuestas de referencia y generadas a conjuntos de palabras únicas en minúsculas
    reference_words = set(reference_response.lower().split())
    generated_words = set(generated_response.lower().split())

    # Precisión: cantidad de palabras correctas en la respuesta generada dividido por el total de palabras generadas
    precision = len(reference_words.intersection(generated_words)) / len(generated_words) if len(generated_words) > 0 else 0

    # Recall: cantidad de palabras correctas en la respuesta generada dividido por el total de palabras de referencia
    recall = len(reference_words.intersection(generated_words)) / len(reference_words) if len(reference_words) > 0 else 0

    # F1-Score: medida armónica entre precisión y recall
    f1_score = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

    return precision, recall, f1_score
  
```

Evaluación de las respuestas se hace sobre el conjunto de datos de validación

El conjunto de datos de **validación** se utiliza para evaluar el rendimiento del modelo después de haber sido entrenado en el conjunto de datos de entrenamiento.

El conjunto de datos de validación se utiliza para medir cómo se generaliza el modelo a datos no vistos durante el entrenamiento y, por lo tanto, proporciona una estimación del rendimiento del modelo en datos del mundo real.

```
# Función para evaluar una instancia y calcular Las métricas
def evaluate_instance2(instance, generated_response):
    # Obtener La pregunta y La respuesta de referencia de La instancia
    question = instance['question']
    reference_response = instance['answers']['text'][0] # Suponiendo que La primera respuesta sea La correcta

    # Calcular Las métricas de precisión, recall y F1-Score
    precision, recall, f1_score = calculate_metrics(reference_response, generated_response)

    # Calcular La precisión (accuracy)
    accuracy = 1 if generated_response.lower() == reference_response.lower() else 0

    return {
        'precision': precision,
        'recall': recall,
        'f1_score': f1_score,
        'accuracy': accuracy
    }
```

```
# Evaluación del conjunto de datos
total_count = len(data['validation'])
precision_total = 0
recall_total = 0
f1_score_total = 0
accuracy_total = 0

# Iterar sobre todas Las instancias del conjunto de datos de validación
for i in tqdm(range(total_count)):
    instance = data['validation'][i]

    # Generar una respuesta para La instancia actual
    generated_response = generate_answer(context, question, model2, tokenizer2, device)

    # Evaluar La instancia y calcular Las métricas
    evaluation_result = evaluate_instance2(instance, generated_response)

    # Agregar Los resultados de precisión, recall, F1-Score y accuracy a Los totales
    precision_total += evaluation_result['precision']
    recall_total += evaluation_result['recall']
    f1_score_total += evaluation_result['f1_score']
    accuracy_total += evaluation_result['accuracy']

# Calcular Las métricas promedio dividiendo La suma total por el número total de instancias
precision_avg = precision_total / total_count
recall_avg = recall_total / total_count
f1_score_avg = f1_score_total / total_count
accuracy_avg = accuracy_total / total_count
```

```
100%|██████████| 102/102 [00:01<00:00, 93.31it/s]
```

```
# Imprime Los resultados
print(f'Precision: {precision_avg}')
print(f'Recall: {recall_avg}')
print(f'F1-Score: {f1_score_avg}')
print(f'Accuracy: {accuracy_avg}')
```

```
Precision: 0.21023965141612175
Recall: 0.11000556883385948
F1-Score: 0.13584312213988756
Accuracy: 0.5686274509803921
```

Función para calcular las métricas de precisión y recall con umbral

- Precision: La proporción de tokens correctos en la respuesta generada dividido por el total de tokens generados.
- Recall: La proporción de tokens correctos en la respuesta generada dividido por el total de tokens de referencia.

Los umbrales de precisión y recall se utilizan para determinar si los valores calculados superan ciertos umbrales predefinidos, y se utilizan para asignar un valor binario (1 o 0) a precisión y recall. Umbral definido arbitrariamente: `threshold = 0.15`

```
def calculate_metrics(reference_response, generated_response):
    # Convertir las respuestas de referencia y generadas a listas de tokens y convertir a minúsculas
    reference_tokens = reference_response.lower().split()
    generated_tokens = generated_response.lower().split()

    # Umbral definido arbitrariamente
    threshold = 0.15

    # Proporción de tokens correctos en la respuesta generada dividido por el total de tokens generados
    precision = len(set(reference_tokens).intersection(generated_tokens)) / len(generated_tokens)

    # Umbral de Precisión
    threshold = 0.15 # Umbral definido arbitrariamente
    umbralPrecision = precision > threshold

    # Precisión (1 si la precisión supera el umbral, de lo contrario 0)
    precision = 1 if umbralPrecision else 0

    # Proporción de tokens correctos en la respuesta generada dividido por el total de tokens de referencia
    recall = len(set(reference_tokens).intersection(generated_tokens)) / len(reference_tokens)

    # Umbral de Recall
    umbralRecall = recall > threshold

    # Recall (1 si el recall supera el umbral, de lo contrario 0)
    recall = 1 if umbralRecall else 0

    # Devuelve las métricas calculadas
    return precision, recall
```

Evaluación de las respuestas se hace sobre el conjunto de datos de validación. Precision y Recall con umbral.

```
# Función para evaluar una instancia y calcular las métricas
def evaluate_instance2(instance, generated_response):
    # Obtener la pregunta y la respuesta de referencia de la instancia
    question = instance['question']
    reference_response = instance['answers']['text'][0] # Suponiendo que la primera respuesta sea la correcta

    # Calcular las métricas de precisión, recall
    precision, recall = calculate_metrics(reference_response, generated_response)

    # Calcular la precisión (accuracy)
    accuracy = 1 if generated_response.lower() == reference_response.lower() else 0

    return {
        'precision': precision,
        'recall': recall,
        'accuracy': accuracy
    }

# Evaluación del conjunto de datos
total_count = len(data['validation'])
precision_total = 0
recall_total = 0
accuracy_total = 0
```



```
inputs = inputs1
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model2.to(device)
inputs = {k: v.to(device) for k, v in inputs.items()}
```

```
# Obtener la salida del modelo
with torch.no_grad():
    output = model2(**inputs)
```

```
# Obtener la respuesta prevista
start_idx = torch.argmax(output.start_logits)
end_idx = torch.argmax(output.end_logits)
```

```
predicted_answer = tokenizer2.convert_tokens_to_string(tokenizer2.convert_ids_to_tokens(inputs['input_ids'])[0][start_idx:end_idx])
```

```
predicted_answer, start_idx, end_idx, start_idx.item(), end_idx.item()
```

```
('aducción, distribución, depuración y alcantarillado',
 tensor(21, device='cuda:0'),
 tensor(29, device='cuda:0'),
 21,
 29)
```

```
instance['answers']
```

```
{'answer_start': [61],
 'text': ['Aducción, distribución, depuración y alcantarillado']}
```

```
given_answer = instance['answers']['text'][0] # Suponiendo que la primera respuesta sea la correcta
given_answer_start = instance['answers']['answer_start'][0]
given_answer, given_answer_start
```

```
('Aducción, distribución, depuración y alcantarillado', 61)
```

```
correct = (predicted_answer.lower() == given_answer.lower())
evaluation = 'Correcto' if correct else f'Incorrecto (Predicción: {predicted_answer}, Solución: {given_answer})'
```

```
print(evaluation)
```

```
Correcto
```

Realizar una inferencia con el modelo entrenado con fine-tuning

Para realizar la inferencia utilizando el modelo cargado `model2`, primero necesitas tokenizar los textos de contexto y pregunta utilizando el tokenizer, luego alimentar los tokens al modelo para obtener las respuestas.

```
# Texto de contexto y pregunta
contexto = "Si ya estás dado de alta en la Oficina Virtual, además de todas las gestiones recogidas en el apartado anterior, puedes realizar las siguientes gestiones:"
pregunta = "¿Qué trámites puedo realizar en relación con mis facturas si estoy registrado en la Oficina Virtual?"
```

```
# Tokenizar el contexto y la pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model2.to(device)
inputs = {k: v.to(device) for k, v in inputs.items()}
```

```
# Realizar la inferencia
device = torch.device("cpu")
```

```
# Realizar la inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))
```

```
# Obtener Los Logits de inicio y fin de La respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en Los Logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar La respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: consultar facturas anteriores, reclamar alguna factura, enviarnos un justificante de pago, solicitar una devolución de saldo y descargar un listado de tus facturas

```
# Texto de contexto y pregunta
contexto = "En tu factura podrás distinguir cuatro conceptos facturados: aducción, distribución, depuración y alcantarillado. Ca
pregunta = "¿Qué conceptos hay en la factura?"

# Tokenizar el contexto y La pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
# Realizar La inferencia
device = torch.device("cpu")

# Realizar La inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Realizar La inferencia
#with torch.no_grad():
#    outputs = model2(**tokens)

# Obtener Los Logits de inicio y fin de La respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en Los Logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar La respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: aducción, distribución, depuración y alcantarillado

```
# Texto de contexto y pregunta
contexto = "En tu factura podrás distinguir cuatro conceptos facturados: aducción, distribución, depuración y alcantarillado. Ca
pregunta = "¿Qué cuatro conceptos hay en la factura?"

# Tokenizar el contexto y La pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
# Realizar La inferencia
device = torch.device("cpu")

# Realizar La inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Obtener Los Logits de inicio y fin de La respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits
```

```
# Encontrar el índice del token con el valor más alto en Los Logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar La respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: aducción, distribución, depuración y alcantarillado

```
pregunta = "¿Cuanto es el tiempo voluntario de pago?"
```

```
# Tokenizar el contexto y la pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
# Realizar La inferencia
device = torch.device("cpu")

# Realizar La inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Obtener Los Logits de inicio y fin de La respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en Los Logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar La respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: 30 días naturales

```
# Texto de contexto y pregunta
contexto = "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino."
pregunta = "¿Qué debo revisar si tengo falta de agua en mi vivienda?"
```

```
# Tokenizar el contexto y la pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
# Realizar La inferencia
device = torch.device("cpu")

# Realizar La inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Obtener Los Logits de inicio y fin de La respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en Los Logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar La respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda)

```
# Texto de contexto y pregunta
contexto = "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua"
pregunta = "¿Qué debo hacer si no tengo agua en mi casa?"

# Tokenizar el contexto y la pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
# Realizar la inferencia
device = torch.device("cpu")

# Realizar la inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Obtener los logits de inicio y fin de la respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en los logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar la respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: en primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. si no, pregunta a algún vecino si tiene agua

```
pregunta = "¿Qué debo hacer si sigo sin agua después de verificar el contador?"

# Tokenizar el contexto y la pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```
# Realizar la inferencia
device = torch.device("cpu")

# Realizar la inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Obtener los logits de inicio y fin de la respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en los logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar la respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)
```

Respuesta: ponte en contacto con nosotros

```
# Texto de contexto y pregunta
contexto = "Puedes elegir siete vías diferentes para contactar con nosotros: La manera más rápida es utilizar el chat de esta Of. de Atención al Cliente."
pregunta = "¿Cuál es el teléfono de atención al cliente?"

# Tokenizar el contexto y la pregunta
tokens = tokenizer2.encode_plus(pregunta, contexto, return_tensors="pt", max_length=512, truncation=True)
```

```

# Realizar La inferencia
device = torch.device("cpu")

# Realizar La inferencia
with torch.no_grad():
    outputs = model2.to(device)(**tokens.to(device))

# Obtener Los Logits de inicio y fin de La respuesta
start_logits = outputs.start_logits
end_logits = outputs.end_logits

# Encontrar el índice del token con el valor más alto en Los Logits de inicio y fin
inicio_respuesta = torch.argmax(start_logits)
fin_respuesta = torch.argmax(end_logits)

# Decodificar La respuesta
respuesta = tokenizer2.decode(tokens['input_ids'][0][inicio_respuesta:fin_respuesta+1])

print("Respuesta:", respuesta)

```

Respuesta: teléfono gratuito 900 365 365

Hacer la inferencia sin contexto, recorriendo el dataset y buscar la pregunta (question) que mas se asemeja

```
from scipy.spatial.distance import cosine
```

```

def predict_answer_select_context(question):
    # Tokenizar La pregunta
    inputs = tokenizer2(question, return_tensors='pt', max_length=512, truncation=True)

    # Buscar La pregunta que más se asemeja
    tokenized_question = inputs['input_ids'][0]
    menor_distancia = 1
    contexto_mas_similar = None

    for example in list(data['train']) + (list(data['validation'])):
        train_question = example['question']
        inputs_train = tokenizer2(train_question, return_tensors='pt', max_length=512, truncation=True)
        tokenized_train_question = inputs_train['input_ids'][0]

        padding_length = abs(len(tokenized_train_question) - len(tokenized_question))
        if len(tokenized_train_question) > len(tokenized_question):
            padded_question = np.pad(tokenized_question, (0, padding_length))
            padded_train_question = tokenized_train_question
        else:
            padded_question = tokenized_question
            padded_train_question = np.pad(tokenized_train_question, (0, padding_length))

        # calcula La distancia del coseno
        cosine_distance = abs(cosine(padded_question, padded_train_question))

        # mientras más cercano a 0 sea La distancia entre Los dos vector mayor será La similitud
        if cosine_distance < menor_distancia:
            menor_distancia = cosine_distance
            contexto_mas_similar = example

    return contexto_mas_similar

```

```
question = '¿Dónde está el contador?'
```

```

contexto_mas_similar = predict_answer_select_context(question)
contexto_mas_similar

```

```
contexto_mas_similar
```

```
{'id': '114',
 'title': 'Preguntas frecuentes: Averías',
 'context': 'En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros.',
 'question': 'Si quiero encontrar el contador, ¿qué debo hacer?',
 'answers': {'answer_start': [384],
 'text': ['En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador']}]}
```

```
contexto_mas_similar['question']
```

```
'Si quiero encontrar el contador, ¿qué debo hacer?'
```

```
contexto_mas_similar['answers']
```

```
{'answer_start': [384],
 'text': ['En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador']}
```

```
contexto_mas_similar['answers']['answer_start'][0]
```

```
384
```

```
contexto_mas_similar['answers']['text'][0]
```

```
'En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador'
```

```
question = "¿Qué puedo avisar al servicio de incidencias?"
```

```
contexto_mas_similar = predict_answer_select_context(question)
contexto_mas_similar['answers']['text'][0]
```

```
'Puedes enviar una comunicación escrita por correo ordinario o fax: Att. Registro General. C/ Santa Engracia, 125 (28003 - Madrid). Horario: de lunes a viernes de 8:30 a 14:00 horas. Fax: 915 451 430'
```

```
question = "¿Cada cuánto tiempo recibo mi factura?"
```

```
contexto_mas_similar = predict_answer_select_context(question)
contexto_mas_similar['answers']['text'][0]
```

```
'Bimestralmente'
```

```
question = "¿Qué conceptos hay en la factura?"
```

```
contexto_mas_similar = predict_answer_select_context(question)
contexto_mas_similar['answers']['text'][0]
```

```
'Aducción, distribución, depuración y alcantarillado'
```

Contexto completo para inferencia

Este código recorre cada ejemplo del conjunto de datos, extrae la pregunta y el contexto de cada ejemplo y los concatena en la variable contexto, separadas por un espacio. Al finalizar este bucle, la variable contexto contendrá el contexto deseado, donde cada pregunta está seguida de su contexto. Se puede utilizar esta variable como contexto en la inferencia con el modelo fine-tuneado.

```
# Inicializar la variable de contexto
contexto = ""

# Construir el contexto con cada pregunta seguida de su respuesta
for example in dataset:
    pregunta = example['question']
    respuesta = example['context']
    contexto += f"{pregunta} {respuesta} "
```

El contexto lo reducimos para Averías y Contacto, para acelerar el ejecución:

```
# Inicializar la variable de contexto
contexto = ""

# Construir el contexto con cada pregunta seguida de su respuesta para los primeros 173 ejemplos
for idx, example in enumerate(dataset):
    if idx >= 173:
        break # Detener el bucle después de los primeros 173 ejemplos
    pregunta = example['question']
    respuesta = example['context']
    contexto += f"{pregunta} {respuesta} "
```

contexto

'¿Qué debo hacer si no tengo agua en mi casa? En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros. ¿Qué pasos debo seguir si no tengo suministro de agua en casa? En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la batería de contadores hay un plano donde podrás encontrar la ubicación de tu contador. Si después de haber hecho estas comprobaciones sigues sin agua, ponte en contacto con nosotros. ¿Qué pasos debo seguir si no tengo acceso a agua en mi vivienda, ¿qué medidas debo tomar? En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún vecino si tiene agua. En caso de que sí tenga, comprueba que la llave de paso interior de tu vivienda esté completamente abierta y verifica que el contador o el equipo de medición tenga las dos llaves de paso abiertas (es decir, giradas completamente a la izquierda). En la bate

pipeline("question-answering")

Usamos la biblioteca Transformers de Hugging Face para responder preguntas con contexto: Para abordar problemas de generación de texto y responder preguntas con contexto, vamos a considerar el uso de bibliotecas de NLP como Transformers de Hugging Face, que proporciona una amplia gama de modelos preentrenados para tareas de generación de texto y preguntas y respuestas.

Este código utiliza el modelo de pregunta-respuesta proporcionado por la biblioteca Transformers de Hugging Face y le proporciona una pregunta junto con el contexto que has definido. Luego, devuelve la respuesta obtenida del modelo.

Puedes experimentar con diferentes modelos y ajustar el contexto según sea necesario para obtener mejores resultados. Además, Hugging Face's Transformers ofrece una amplia documentación y ejemplos que pueden ayudarte a entender mejor cómo utilizar sus modelos para diferentes tareas de procesamiento de lenguaje natural.

```
from transformers import pipeline
```

En la condición `if result["score"] > 0.79`, se verifica si el score de confianza de la respuesta supera 0.79 antes de devolver la respuesta. Esto puede ser útil para filtrar respuestas potencialmente incorrectas. Sin embargo, hay que tener en cuenta que este valor puede necesitar ser ajustado según las necesidades y la precisión del modelo.

Los modelos de lenguaje natural son capaces de procesar texto en cualquier formato de capitalización. Sin embargo, hay algunos casos en los que podría influir:

1. Coincidencia exacta: Si el texto de la pregunta y el texto del contexto son idénticos en términos de capitalización, es más probable que se encuentre una coincidencia exacta y la respuesta se encuentre más fácilmente.
2. Preprocesamiento de texto: Algunos modelos o pipelines de procesamiento de lenguaje natural pueden realizar preprocesamientos específicos en el texto antes de procesarlo. Si el pipeline realiza algún tipo de normalización de texto que convierte todo el texto en minúsculas, entonces podría haber una diferencia en el procesamiento de las preguntas en mayúsculas y minúsculas.

Para asegurar de que las preguntas en mayúsculas se procesen correctamente, se pueden convertir todas las preguntas a minúsculas antes de pasarlas al modelo.

```
# def answer_question(question):
#     Convertir la pregunta a minúsculas
#     question = question.lower()
#     Obtener la respuesta del modelo
#     result = nlp(question=question, context=context)
```

De esta manera, independientemente de si el usuario escribe la pregunta en mayúsculas o minúsculas, será procesada de la misma manera. Esto puede ayudar a mejorar la consistencia en la búsqueda de respuestas.

```
# Cargar el modelo de pregunta-respuesta
nlp = pipeline("question-answering", model=model12, tokenizer=tokenizer2)
```

```
def answer_question(question):
    """
    Función para responder preguntas dadas una pregunta y un contexto predefinido.
    """
    # Obtener la respuesta del modelo
    result = nlp(question=question, context=contexto)

    if result["score"] > 0.79:
        # Verificar si se encontró una respuesta
        if result['answer']: # Si la respuesta no está vacía
            # Retornar la respuesta encontrada
            return result['answer']
        else:
            # Devolver un mensaje indicando que la pregunta debe ser reformulada
            return "Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta sobre Averías o"

    else:
        # Devolver un mensaje indicando que la pregunta debe ser reformulada
        return "No supero el 0.79 score. Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pre"
```

```
pregunta = "¿Qué cuatro conceptos hay en la factura?"
```

```
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.5704760551452637,
 'start': 84774,
 'end': 84836,
 'answer': 'indicar tu número de teléfono y el rango horario que prefieras'}
```

```
resp["score"]
```

```
0.5704760551452637
```

Visualización del resultado en formato HTML

```
from IPython.core.display import HTML
```

```
if resp["score"] > 0.79:
    answer = resp["answer"]
    display(HTML(f'<h2>{pregunta.upper()}</h2>'))
    marked_text = str(contexto.replace(answer, f"<mark>{answer}</mark>"))
    display(HTML(f'""<blockquote> {marked_text} </blockquote>""'))
else:
    answer = "No supero el 0.79 score. Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta."
    display(HTML(f'<h2>{pregunta.upper()}</h2>'))
    display(HTML(f'<h2>{answer}</h2>'))
```

¿QUÉ CUATRO CONCEPTOS HAY EN LA FACTURA?

No supero el 0.79 score. Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta sobre Averías o Contacto

```
pregunta = " "
```

```
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.02324775978922844,
 'start': 90090,
 'end': 90119,
 'answer': 'teléfono gratuito 900 365 365'}
```

Visualización del resultado en formato HTML

```
if resp["score"] > 0.79:
    answer = resp["answer"]
    display(HTML(f'<h2>{pregunta.upper()}</h2>'))
    marked_text = str(contexto.replace(answer, f"<mark>{answer}</mark>"))
    display(HTML(f'""<blockquote> {marked_text} </blockquote>""'))
else:
    answer = "No supero el 0.79 score. Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta."
    display(HTML(f'<h2>{pregunta.upper()}</h2>'))
    display(HTML(f'<h2>{answer}</h2>'))
```

No supero el 0.79 score. Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta sobre Averías o Contacto

```
pregunta = "¿A qué número puedo llamar para reportar una incidencia en el suministro de agua?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.9807830452919006,
 'start': 54090,
 'end': 54101,
 'answer': '900 365 365'}
```

Visualización del resultado en formato HTML

```

: if resp["score"] > 0.79:
    answer = resp["answer"]
    display(HTML(F'<h2>{pregunta.upper()}</h2>'))
    marked_text = str(contexto.replace(answer, f"<mark>{answer}</mark>"))
    display(HTML(F'""<blockquote> {marked_text} </blockquote>""'))
else:
    answer = "No supero el 0.79 score. Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta."
    display(HTML(F'<h2>{pregunta.upper()}</h2>'))
    display(HTML(F'<h2>{answer}</h2>'))

```

solicitud, así como un teléfono de contacto. ¿A qué dirección de correo electrónico puedo escribir para comunicarme con Canal de Isabel II? Puedes elegir siete vías diferentes para contactar con nosotros: La manera más rápida es utilizar el chat de esta Oficina Virtual. Si prefieres hablar por teléfono, llama a nuestro teléfono gratuito 900 365 365 en horario de 08:00 a 20:00. Si necesitas comunicar una incidencia o avería, el servicio se encuentra operativo las 24 horas del día, los 365 días del año. Otra opción telefónica es que te llamemos nosotros. Solo tienes que indicar tu número de teléfono y el rango horario que prefieras. Puedes utilizar un formulario de contacto. Tienes la posibilidad de escribirnos un correo electrónico a la dirección: clientes@canaldeisabelsegunda.es. Puedes pedir cita para que te atendamos en nuestros centros de atención al cliente. Y si prefieres métodos tradicionales, puedes enviar una comunicación escrita por correo ordinario o fax: Att. Registro General. C/ Santa Engracia, 125 (28003 - Madrid). Horario: de lunes a viernes de 8:30 a 14:00 horas. Fax: 915 451 430. Independientemente del canal de comunicación que prefieras utilizar, no olvides indicarnos como referencia tu número de contrato o la dirección de la finca objeto de la solicitud, así como un teléfono de contacto. ¿Cuál es la dirección de correo electrónico para contactar con el Canal de Isabel II? Puedes elegir siete vías diferentes para contactar con nosotros: La manera más rápida es utilizar el chat de esta Oficina Virtual. Si prefieres hablar por teléfono, llama a nuestro teléfono gratuito 900 365 365 en horario de 08:00 a 20:00. Si necesitas comunicar una incidencia o avería, el servicio se encuentra operativo las 24 horas del día, los 365 días del año. Otra opción telefónica es que te llamemos nosotros. Solo tienes que indicar tu número de teléfono y el rango horario que prefieras. Puedes utilizar un formulario de contacto. Tienes la posibilidad de escribirnos un correo electrónico a la dirección: clientes@canaldeisabelsegunda.es. Puedes pedir cita para que te atendamos en nuestros centros de atención al cliente. Y si prefieres métodos tradicionales, puedes enviar una comunicación escrita por correo ordinario o fax: Att. Registro General. C/ Santa Engracia, 125 (28003 - Madrid). Horario: de lunes a

Inferencia con pipeline("question-answering"), para preguntas que entran en nuestro contexto

```

: pregunta = "¿Cuál es la dirección de correo electrónico de Canal de Isabel II?"
resp = nlp(context=contexto, question=pregunta)
resp

```

```

: {'score': 0.9735689163288008,
  'start': 85979,
  'end': 86025,
  'answer': 'la dirección: clientes@canaldeisabelsegunda.es'}

```

```

: pregunta = "¿Cuál es el contacto del Canal de Isabel II?"
resp = nlp(context=contexto, question=pregunta)
resp

```

```

: {'score': 0.9396594762802124,
  'start': 74653,
  'end': 74699,
  'answer': 'la dirección: clientes@canaldeisabelsegunda.es'}

```

```

: pregunta = "¿Cuál es el teléfono del servicio de notificaciones de incidencias en el suministro de agua?"
resp = nlp(context=contexto, question=pregunta)
resp

```

```

: {'score': 0.8074807524681091,
  'start': 34744,
  'end': 34755,
  'answer': '900 365 365'}

```

```
pregunta = "¿Qué puedo avisar al servicio de incidencias?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.9267107248306274,
 'start': 34848,
 'end': 34870,
 'answer': 'cortes de agua o fugas'}
```

```
pregunta = "¿Cuál es el teléfono de atención al cliente?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.7625647783279419,
 'start': 74249,
 'end': 74260,
 'answer': '900 365 365'}
```

Inferencia con pipeline("question-answering"), para preguntas que "No" entran en nuestro contexto

Ya que vamos a limitar al asistente a preguntas relacionadas con Averías o Contacto. Comprobamos el score de alguna pregunta relacionada con el Canal de Isabel II, pero con otra temática. Así podemos evaluar un score que nos asegure buenas respuestas.

```
pregunta = "¿En qué horario puedo contactarlos?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.8467395305633545,
 'start': 128831,
 'end': 128909,
 'answer': 'Solo tienes que indicar tu número de teléfono y el rango horario que prefieras'}
```

```
pregunta = "¿Cada cuánto tiempo recibo mi factura?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.8981596827507019,
 'start': 85130,
 'end': 85177,
 'answer': 'comunicación escrita por correo ordinario o fax'}
```

```
pregunta = "¿Qué trámites puedo realizar en relación con mis facturas si estoy registrado en la Oficina Virtual?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.9139502644538879,
 'start': 75844,
 'end': 75909,
 'answer': 'puedes enviar una comunicación escrita por correo ordinario o fax'}
```

```
pregunta = "¿Qué cuatro conceptos hay en la factura?"
resp = nlp(context=contexto, question=pregunta)
resp
```

```
{'score': 0.5704760551452637,
 'start': 84774,
 'end': 84836,
 'answer': 'indicar tu número de teléfono y el rango horario que prefieras'}
```

Transformers de Hugging Face para realizar inferencias de preguntas y respuestas

Se utiliza la biblioteca Transformers de Hugging Face para realizar inferencias de preguntas y respuestas utilizando el modelo preentrenado con fine-tuning que ha sido entrenado con datos etiquetados de preguntas y respuestas del Canal de Isabel II.

1. Carga del Modelo y Tokenizador: Utiliza la función `AutoModelForQuestionAnswering.from_pretrained()` para cargar un modelo preentrenado para la tarea de pregunta-respuesta, y `AutoTokenizer.from_pretrained()` para cargar el tokenizador asociado con el modelo. Ambos son cargados a partir del nombre del modelo especificado en la variable `model_name`.
2. Configuración del Pipeline: Se configura un pipeline de pregunta-respuesta utilizando la función `pipeline('question-answering', model=model, tokenizer=tokenizer)`. Este pipeline utiliza el modelo y el tokenizador cargados previamente.
3. Definición del Contexto: Se define un contexto largo que contiene información sobre un tema específico. Este contexto se utiliza como fuente para buscar respuestas a las preguntas.
4. Función de Respuesta a Preguntas: La función `answer_question(question)` toma una pregunta como entrada, y utiliza el pipeline previamente configurado para encontrar la respuesta en el contexto proporcionado. La respuesta encontrada se devuelve como resultado de la función.
5. Ejecución de la Función con una Pregunta Específica: Se proporcionan preguntas específicas ("¿Cada cuánto tiempo recibo mi factura?", "¿Cuanto es el tiempo voluntario de pago?") a la función `answer_question()`. Estas preguntas se pasan al modelo, junto con el contexto definido anteriormente, y el modelo devuelve la respuesta encontrada en el contexto.

En resumen, este código utiliza un modelo preentrenado para responder preguntas dadas un contexto específico, sobre los servicios de agua en la empresa Canal de Isabel II de suministro de agua.

Modelo preentrenado completo para responder preguntas dadas un contexto de Avería y Contacto, sobre los servicios de agua en la empresa Canal de Isabel II de suministro de agua

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline
```

```
# Cargar el modelo y el tokenizador
model_name = 'Antonio49/ModeloCanal'
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
# Configurar el pipeline para la pregunta-respuesta
nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)
```

```
# Contexto integrado en el código
context = contexto
```

```
Some weights of the model checkpoint at Antonio49/ModeloCanal were not used when initializing BertForQuestionAnswering: ['qa_outputs.additional_layers.0.bias', 'qa_outputs.additional_layers.0.weight', 'qa_outputs.additional_layers.3.bias', 'qa_outputs.additional_layers.3.weight']
```

```
- This IS expected if you are initializing BertForQuestionAnswering from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForQuestionAnswering from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
```

```
def answer_question(question):
    """
    Función para responder preguntas dadas una pregunta y un contexto predefinido.
    """
    # Obtener la respuesta del modelo
    result = nlp(question=question, context=context)

    # Verificar si se encontró una respuesta
    if result['answer']: # Si la respuesta no está vacía
        # Retornar la respuesta encontrada
        return result['answer']
    else:
        # Devolver un mensaje indicando que la pregunta debe ser reformulada
        return "Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta"
```

```
def answer_question(question):
    """
    Función para responder preguntas dadas una pregunta y un contexto predefinido.
    """
    # Obtener La respuesta del modelo
    result = nlp(question=question, context=contexto)

    if result["score"] > 0.79:
        # Verificar si se encontró una respuesta
        if result['answer']: # Si La respuesta no está vacía
            # Retornar La respuesta encontrada
            return result['answer']
        else:
            # Devolver un mensaje indicando que La pregunta debe ser reformulada
            return "Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta"
    else:
        # Devolver un mensaje indicando que La pregunta debe ser reformulada
        return "Lo siento, no score"

# Prueba de La función
print(answer_question("¿Qué puedo avisar al servicio de incidencias?"))
```

cortes de agua o fugas

```
def answer_question(question):
    """
    Función para responder preguntas dadas una pregunta y un contexto predefinido.
    """
    # Obtener La respuesta del modelo
    result = nlp(question=question, context=contexto)

    if result["score"] > 0.79:
        # Verificar si se encontró una respuesta
        if result['answer']: # Si La respuesta no está vacía
            # Retornar La respuesta encontrada
            return result['answer']
        else:
            # Devolver un mensaje indicando que La pregunta debe ser reformulada
            return "Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta"
    else:
        # Devolver un mensaje indicando que La pregunta debe ser reformulada
        return "Lo siento, no score"

# Prueba de La función
print(answer_question("¿Qué cuatro conceptos hay en la factura?"))
```

Lo siento, no score

Modelo preentrenado reducido (prueba) para responder preguntas dadas un contexto específico reducido (prueba), sobre los servicios de agua en la empresa Canal de Isabel II de suministro de agua

```
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline

# Cargar el modelo y el tokenizador
model_name = 'Antonio49/Personal'
model = AutoModelForQuestionAnswering.from_pretrained(model_name)
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Configurar el pipeline para La pregunta-respuesta
nlp = pipeline('question-answering', model=model, tokenizer=tokenizer)

# Contexto integrado en el código
context = "En primer lugar, averigua si la falta de agua afecta a todos los puntos de agua de tu casa. Si no, pregunta a algún ve
```

```
def answer_question(question):
    """
    Función para responder preguntas dadas una pregunta y un contexto predefinido.
    """
    # Obtener La respuesta del modelo
    result = nlp(question=question, context=context)

    # Verificar si se encontró una respuesta
    if result['answer']: # Si La respuesta no está vacía
        # Retornar La respuesta encontrada
        return result['answer']
    else:
        # Devolver un mensaje indicando que La pregunta debe ser reformulada
        return "Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta"
```

En este código, la función verificará si la respuesta devuelta por el modelo no está vacía. Si la respuesta está vacía, la función devolverá un mensaje indicando que no pudo encontrar una respuesta para la pregunta.

```
question = "¿Cada cuánto tiempo recibo mi factura?"
answer_question(question)
```

```
'cada dos meses'
```

```
question = "¿Cuanto es el tiempo voluntario de pago?"
answer_question(question)
```

```
'30 días naturales'
```

```
question = "¿Cuál es la dirección de correo electrónico de Canal de Isabel II?"
answer_question(question)
```

```
'900 365 365'
```

Aquí probamos si supera un nivel de score

```
def answer_question(question):
    """
    Función para responder preguntas dadas una pregunta y un contexto predefinido.
    """
    # Obtener La respuesta del modelo
    result = nlp(question=question, context=context)

    if result["score"] > 0.2:
        # Verificar si se encontró una respuesta
        if result['answer']: # Si La respuesta no está vacía
            # Retornar La respuesta encontrada
            return result['answer']
        else:
            # Devolver un mensaje indicando que La pregunta debe ser reformulada
            return "Lo siento, no pude encontrar una respuesta para tu pregunta. Por favor, reformula tu pregunta"

    else:
        # Devolver un mensaje indicando que La pregunta debe ser reformulada
        return "Lo siento, no score"
```

```
question = "¿Cuanto es el tiempo voluntario de pago?"
answer_question(question)
```

```
'30 días naturales'
```

```
resp = nlp(context=context, question=question)
resp
```

```
{'score': 0.20767243206501007,
 'start': 7623,
 'end': 7640,
 'answer': '30 días naturales'}
```

```
question = "¿Que puedes responderme?"
answer_question(question)
```

```
'Lo siento, no score'
```

```
resp = nlp(context=context, question=question)
resp
```

```
{'score': 0.05990104749798775,
 'start': 8998,
 'end': 9040,
 'answer': 'Estudiamos individualmente las solicitudes'}
```

```
question = " "
answer_question(question)
```

```
'Lo siento, no score'
```

```
resp = nlp(context=context, question=question)
resp
```

```
{'score': 0.0022760480642318726,
 'start': 1485,
 'end': 1492,
 'answer': 'Durante'}
```

```
question = "¿Cada cuánto tiempo recibo mi factura?"
answer_question(question)
```

```
'Lo siento, no score'
```

```
resp = nlp(context=context, question=question)
resp
```

```
{'score': 0.15548183023929596,
 'start': 7539,
 'end': 7553,
 'answer': 'cada dos meses'}
```