Francisco Acuña - 20220565

Análisis del Laboratorio: Modelos de Lenguaje

Comparación de Modelos

Se detalla el desempeño de cada modelo, basándose en la información de la página HuggingFace y los resultados observados en la práctica.

a. Tiempo de Respuesta

El tiempo de respuesta de un modelo se mide en dos fases: el tiempo de preparación (carga) y el tiempo para generar la respuesta (inferencia).

- Tiempo de Preparación:
  - Qwen fue el más rápido en cargar (6.07 segundos), lo cual es consistente con ser el modelo más pequeño.
  - Doctor-Shotgun tardó un tiempo de carga esperado para su tamaño (10.12 segundos).
  - El modelo TinyLlama requirió un tiempo considerablemente mayor para prepararse (48.23 segundos). Esto probablemente sucedió porque es un modelo "base", el cual no está optimizado para responder preguntas directamente y necesita más tiempo para iniciar el proceso.
- Tiempo para Generar la Respuesta:
  - Doctor-Shotgun fue el más rápido generando la respuesta (4.49 segundos), aunque también produjo el texto más corto.
  - TinyLlama tuvo una velocidad de inferencia intermedia (9.67 segundos).
  - Qwen fue el más lento (11.38 segundos), a pesar de ser el modelo más pequeño. Es posible que su arquitectura interna requiera un proceso más complejo para generar cada palabra.

b. Uso de Recursos de Cómputo

Este es un factor muy importante, sobre todo cuando no se dispone de un equipo de alta potencia.

- Parámetros (Tamaño y Complejidad del Modelo):
  - Qwen: Es el modelo más "liviano", con aproximadamente 500 millones de parámetros, lo que lo hace adecuado para equipos con menos potencia.
  - TinyLlama: Es un modelo de tamaño mediano, con 1.1 mil millones de parámetros.
  - Doctor-Shotgun: También cuenta con 1.1 mil millones de parámetros, ya que es una versión mejorada y afinada del TinyLlama original.
- Uso de la Computadora (Memoria de la Tarjeta de Video):
  - El factor clave para la ejecución fue el uso de la "cuantización a 4 bits" (USE_4BIT = True). Esta técnica funciona de manera similar a comprimir un archivo para que ocupe menos espacio. Gracias a esto, fue posible cargar los modelos grandes en la memoria limitada de la tarjeta de video (GPU) de Google Colab. Sin esta técnica, la ejecución

no habría sido posible.

c. Calidad de las Respuestas

La calidad de la respuesta no depende únicamente del tamaño del modelo, sino fundamentalmente de su entrenamiento y afinamiento.

- TinyLlama: Resultado nulo. Al ser un modelo "base", no está diseñado para conversar o seguir instrucciones. El modelo solo repitió el texto de entrada (prompt), sin comprender la instrucción.
- Doctor-Shotgun: Respuesta de baja calidad. Aunque está entrenado para seguir instrucciones, el modelo alucinó la respuesta. Confundió el término "Transformer" de inteligencia artificial con un transformador eléctrico y, además, la descripción proporcionada era incorrecta.
- Qwen: Respuesta de baja calidad. Similar al caso anterior, el modelo se confundió de contexto y generó una explicación sin coherencia sobre un transformador eléctrico.
- ChatGPT: Respuesta de alta calidad. Fue el único modelo que interpretó correctamente el contexto de la pregunta. Explicó el concepto de "Transformer" de IA de forma correcta y con ejemplos adecuados, cumpliendo con lo solicitado.

Ficha Técnica del Laboratorio

Programas y Librerías Utilizadas

Las librerías clave que hicieron posible la ejecución del laboratorio fueron:

- torch: Es la librería base para inteligencia artificial. Se encarga de los cálculos numéricos complejos y de gestionar el uso de la tarjeta de video.
- transformers: La herramienta principal de Hugging Face. Permite descargar y utilizar los modelos probados de una manera muy sencilla.
- accelerate: Una librería auxiliar que optimiza la ejecución del código para que se adapte mejor al hardware disponible y funcione más rápido.
- bitsandbytes: Esta fue la librería que permitió la compresión de los modelos para reducir su uso de memoria. Esta librería fue indispensable para la ejecución del laboratorio.

Funciones Principales del Código

Dentro del código, se utilizaron tres funciones principales:

- AutoTokenizer.from_pretrained(): Esta función toma el texto de entrada y lo convierte en números, un formato que el modelo puede entender.
- AutoModelForCausalLM.from_pretrained(): Esta función descarga y carga el modelo (su "cerebro") en la memoria de la computadora.
- pipeline(): Se trata de una función de alto nivel que simplifica todo el proceso, creando una interfaz fácil para solicitar una respuesta al modelo sin necesidad de programar cada paso

individualmente.

Ajustes Clave para la Ejecución

Para asegurar el correcto funcionamiento, se realizaron algunos ajustes importantes:

- model = "...": Este parámetro se usó para indicar al programa el nombre exacto del modelo de Hugging Face que se deseaba probar.
- load_in_4bit=True: Este fue el ajuste más importante. Le indicó al programa que aplicara la compresión (cuantización) al modelo para reducir su tamaño y permitir que se cargara en la memoria disponible.
- device_map="auto": Este ajuste instruye al programa para que utilice la tarjeta de video (GPU) de forma automática si se encuentra una disponible, lo que acelera significativamente el proceso.
- max_new_tokens=200: Con este parámetro se estableció un límite a la longitud de la respuesta, para evitar que el modelo generará texto de forma infinita