INTELIGENCIA ARTIFICIAL

Profesor:

Alcaraz Chavez Jesus Eduardo

Nombre:

Kevin Alejandro Cuevas Crisantos

Numero de Control:

21121503

Proyecto 4

Informe Técnico proyecto 4

Fine-Tuning de Llama 3.2 para Tutoría de Algoritmos

1. Introducción

El presente proyecto tiene como objetivo desarrollar un **sistema de tutoría inteligente especializado en algoritmos fundamentales**, específicamente **Bubble Sort, Búsqueda Binaria y Merge Sort**, utilizando un Modelo Grande de Lenguaje (LLM) ajustado mediante **Fine-Tuning eficiente**.

Una de las principales limitaciones abordadas en este trabajo es el **uso de hardware de consumo**, concretamente una GPU **RTX 3050 con 4GB de VRAM**, lo cual obligó a emplear técnicas modernas de optimización de memoria y entrenamiento eficiente.

El resultado final es un tutor confiable, determinista y libre de alucinaciones, diseñado para responder **únicamente** sobre los algoritmos entrenados, garantizando fidelidad total en las respuestas.

2. Objetivo del Proyecto

- Especializar un modelo LLM moderno en **algoritmos básicos de ordenamiento y búsqueda**.
- Garantizar **respuestas exactas**, incluyendo código correcto y explicaciones rigurosas.
- Evitar respuestas fuera del dominio (alucinaciones).
- Lograr todo el proceso en **hardware doméstico limitado**.

3. Arquitectura General del Sistema

El sistema se compone de tres bloques principales:

1. **Preparación del Dataset (data_prep.py)**
2. **Entrenamiento con Fine-Tuning (train.py)**
3. **Inferencia Controlada (inference.py)**

Cada componente cumple una función específica para asegurar precisión, especialización y control del comportamiento del modelo.

4. Modelo Base

- **Modelo:** meta-llama/Llama-3.2-1B-Instruct
- **Tamaño:** 1 billón de parámetros

Justificación

Se eligió esta variante debido a que:

- Representa el **estado del arte en modelos ligeros**.
- Mantiene buena coherencia gramatical.
- Es viable para ejecución y entrenamiento local.
- Permite especialización mediante técnicas PEFT sin necesidad de grandes recursos.

5. Técnica de Entrenamiento: QLoRA

Para permitir el ajuste fino en una GPU de 4GB, se implementó **QLoRA (Quantized Low-Rank Adaptation)**.

5.1 Cuantización de 4 bits (NF4)

- El modelo base se carga en **4 bits** usando bitsandbytes.
- Reduce el consumo de memoria a aproximadamente **1.5 GB**.
- Mantiene estabilidad numérica gracias a NF4.

5.2 Adaptadores LoRA

- No se reentrena todo el modelo.
- Se insertan **adaptadores pequeños** en las capas:
    - q_proj
    - v_proj
- Solo se entrena aproximadamente **el 1% de los parámetros totales**.

Esto reduce drásticamente el costo computacional sin sacrificar desempeño en el dominio objetivo.

6. Estrategia de Sobreajuste Controlado

A diferencia de modelos generalistas, este proyecto **busca intencionalmente el sobreajuste**.

¿Por qué sobreajustar?

- El dominio es **muy pequeño y bien definido**.

- Se requieren **respuestas exactas**, no aproximadas.
- El modelo debe comportarse como un **tutor académico**, no creativo.

Implementación

- **Dataset sintético curado** con ejemplos perfectos.
- Varias formulaciones de preguntas que conducen a la **misma respuesta**.
- Inclusión de **ejemplos negativos** para reforzar rechazos.

Hiperparámetros clave

- Épocas: Entrenamiento intensivo
- Learning Rate: 2e-4

Resultado

El modelo actúa como:

- Un **razonador** para entender la intención del usuario.
- Una **base de datos semántica** que recupera respuestas exactas.

Esto elimina prácticamente el fenómeno de alucinaciones.

7. Preparación del Dataset (data_prep.py)

El script data_prep.py genera automáticamente el dataset de entrenamiento.

7.1 Contenido del Dataset

Para cada algoritmo:

- Explicación general
- Complejidad computacional
- Pasos detallados
- Código Python estándar
- Simulaciones paso a paso

7.2 Variación de Preguntas

Se generan múltiples variantes lingüísticas para una misma respuesta:

- "¿Qué es Bubble Sort?"
- "Explícame cómo funciona Bubble Sort"
- "Resumen de Bubble Sort"

Esto entrena al modelo a **generalizar la intención**, pero **responder de forma idéntica**.

7.3 Ejemplos Negativos

Para cualquier tema fuera del dominio:

- Quick Sort
- Grafos
- Machine Learning
- Bases de datos

El modelo aprende a responder **exactamente**:

"Lo siento, solo tengo conocimientos sobre los algoritmos: Bubble Sort, Búsqueda Binaria y Merge Sort. No puedo ayudarte con otros temas."

8. Formato del Dataset

Cada ejemplo se construye con un **System Prompt estricto**:

- Define explícitamente los límites de conocimiento.
- Obliga al modelo a rechazar temas no permitidos.

El texto final se guarda en formato JSONL y se utiliza directamente para entrenamiento supervisado.

9. Entrenamiento del Modelo (train.py)

El archivo train.py gestiona todo el proceso de Fine-Tuning.

9.1 Optimizaciones de Hardware

- **Gradient Accumulation (4 pasos):** simula batch size mayor.
- **FP32 forzado:** evita errores de drivers en Windows.
- **Gradient Checkpointing:** reduce uso de memoria.

9.2 Librerías Utilizadas

- transformers
- datasets
- peft
- trl
- bitsandbytes

9.3 Proceso

1. Carga del modelo base cuantizado
2. Inserción de adaptadores LoRA
3. Entrenamiento supervisado (SFT)
4. Guardado del modelo ajustado

10. Inferencia Determinista (inference.py)

A diferencia de un chatbot tradicional, este sistema:

- No genera texto libre
- No improvisa respuestas

Funcionamiento

1. Se carga el dataset entrenado
2. La entrada del usuario se normaliza
3. Se busca una coincidencia semántica
4. Se devuelve la **respuesta exacta** del dataset
5. Si no hay coincidencia → respuesta de rechazo

Esto garantiza:

- 100% de fidelidad
- Comportamiento predecible
- Uso académico confiable

11. Ventajas del Sistema

- Entrenamiento en hardware doméstico
- Especialización extrema
- Sin alucinaciones
- Código y explicaciones exactas
- Ideal para uso educativo

12. Conclusión

Este proyecto demuestra que es totalmente viable **especializar modelos de lenguaje modernos en dominios técnicos estrictos** utilizando recursos limitados.

Mediante el uso de **QLoRA, PEFT y sobreajuste controlado**, se logró construir un tutor inteligente capaz de responder con **precisión absoluta**, comportamiento determinista y sin errores conceptuales.

El sistema valida que los LLMs no solo pueden ser generadores de texto, sino también **herramientas educativas confiables**, cuando se entrenan con una arquitectura y estrategia adecuadas.