Relazione progetto di Deep learning

Git link: https://github.com/Saveriucci/Progetto-deep-Learning

L'obiettivo del progetto è addestrare alcuni Small Language Models (SLM) a estrarre informazioni rilevanti da testi in linguaggio naturale e convertirle in un formato rigido e leggibile dalle macchine, in particolare JSON.

Il progetto mira a dimostrare che la specializzazione del modello attraverso fine-tuning può superare la generalizzazione su task verticali. In particolare, mostra come i modelli compatti, opportunamente adattati, possano raggiungere le prestazioni di Large Language Model (LLM) molto più grandi, riducendo significativamente i costi computazionali.

Small Language Models:

- Phi-3 (3,5 B parametri):
  - Adatto per questo task → Ottime performances/velocità
  - Instruct è la variante su cui fare fine tuning
- Mistral (7 B parametri):
  - Instruct è la variante su cui fare fine tuning
  - Alta efficienza di inferenza → Efficiente
  - Buone capacità linguistiche generali
  - Forte su task di estrazione e generazione compatta
- Qwen 2.5 (1,5 B parametri):
  - Instruct è la variante su cui fare fine tuning
  - Ottimizzato per l'instruction-following, output strutturati (ad esempio, JSON, tabelle), la codifica e la risoluzione di problemi matematici

Struttura instruct:

- Token embedding: mapping token in vettori (Parametro: W_{embed} vocab_size X hidden_size)
- Partial Encoding: rotary (Parametri: Q, K non addestrabili)
- Masked self attention: attenzione causale (il token al tempo t vede solo tempi<t) (Parametri: W_Q, W_K, W_V per attenzione, W_O per l'output)
- Feed Forward Network (Parametri: $W_1 \in \mathbb{R}^{d \times dgg}, W_2 \in \mathbb{R}^{d \times dgg}, W_3 \in \mathbb{R}^{dgg \times d}$)
- Residual Connection + Layer Norm (Parametri: scale r, shift β)

Blocco che si ripete i volte

- Large Modeling Head (Parametro: W_{ln} hidden_size X vocab_size)

Step principali:

1) Punto di partenza: SLM Instruct
2) Cosa si osserva prima: Comportamento base SLM su dataset Recipe
3) Cosa si osserva dopo: Comportamento SLM fine tuned
   - Il fine tuning viene effettuato cambiando i pesi del modello per renderlo migliore

- Non andiamo a cambiare la struttura del modello, quella rimane così come si presenta
- Si cambia quindi la parte relativa ai parametri, quindi bisogna capire chi ha il controllo
    - Cosa entra come input
    - Cosa esce come output
    - Quanto rigido deve essere l'output

    Questo consente al SLM di capire in che modo la loss varia, vengono forniti input e output e il SLM si adatta, propagando i gradienti, in modo da ottenere i risultati desiderati.

Perciò ci sono 3 fasi:

1. **Inferenza:** viene dato l'input ⟶ il modello genera l'output (senza aggiornare i pesi)
2. **Addestramento/Fine Tuning:** il modello viene addestrato su coppe (input, output desiderato) ⟶ produce una previsione ⟶ loss confronta previsione con output desiderato ⟶ i pesi vengono aggiornati
3. **Valutazione:** viene dato l'input ⟶ confronto output

Dataset:

- Recipe NLG (2 milioni di ricette)
- Formato iniziale: {Id, Title, Ingredients, Directions, Link, Source, NER}
- Formato che utilizzeremo: testo scritto in modo complesso senza indicare esplicitamente titolo e procedure in modo che SLM debba ricavarli da solo (useremo le prime 600/1000 ricette)

Fase 1: Inferenza

- 50 ricette per ogni SLM (ovviamente le stesse)
- Si confrontano i risultati

Fase 2: Addestramento

- Impossibile svolgerlo Full Fine Tuning perché richiederebbe troppa memoria (circa 40GB per mantenere pesi e gradienti), per cui optiamo per LoRa (Low Rank Adaptation) e QLoRA (LoRA quantizzato)
- Viene effettuato sul dataset ristretto contenente 1000 ricette escludendo però le prime 50 (usate per l'inferenza) e le ultime 100 (riservate alla valutazione)
- **LoRA:**
    - Tecnica di fine tuning che permette di adattare un grande modello linguistico ad un compito specifico senza riaddestrare tutti i suoi parametri
    - Invece di modificare direttamente i parametri si apprendono piccole correzioni aggiuntive che si sommano ai pesi originali
    - Si ha la matrice dei pesi originali W, LoRa introduce la variazione appresa ΔW

- k= dimensione input, d=dimensione output, r=rango (iperparametro, controllo il numero di parametri allenabili), $r \ll \min(d, k)$
- $A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times k}, \Delta W = A \cdot B$ quindi vengono aggiornate solo A e B, $W' = W + \Delta W$
- A = Proiezione spazio di LoRA in spazio di uscita, inizializzata a 0 (r->d): consente di vedere come queste direzioni influenzano l'output
- B = Proiezione dello spazio di input nello spazio di LoRA, inizializzata con valori cauali (k->r): consente di vedere quali direzioni dello spazio di input sono rilevanti per il nuovo task

Fase 3: Valutazione

- Dopo l'addestramento si danno in pasto a SLM le 50+100 ricette che non erano state usate nella fase 2 e si valutano i risultati
- Possibili metriche di valutazione:
  L1. Percentuale di JSON sintatticamente validi
  L2. Percentuale di JSON che rispettano lo schema
  L3. Percentuale di JSON semanticamente corretti

LLM (Large Language Model) Baseline:

- Diamo in pasto le 150 ricette di valutazione ad un LLM (più di 15 miliardi di parametri)
- Possibili LLM:
  - Gemini 25 Pro
  - Command R+ 104B (Paragonabile a GPT-4)
  - Llama 3.3 70B ← Nostra scelta

Risultati:

SLM/LLM	Parametri	L1 inferenza	L1 valutazione	L2 inferenza	L2 valutazione	L3 inferenza	L3 valutazione
Qwen 2.5	1.5 B	88% (44/50)	100%	88% (44/50)	100%	86% (43/50)	96% (144/150)
Phi3	3.5 B	100%	100%	92% (46/50)	100%	96% (48/50)	97% (145/150)
Mistral	7 B	100%	100%	100%	100%	98% (49/50)	98,7% (148/150)
Llama 3.3	70 B	100%	—	100%	—	100%	—

I valori sono coerenti con le aspettative: miglioramento evidente su Qwen, ma anche gli altri SLM mostrano un miglioramento meno evidente viste le già alte performances in fase di inferenza.