**GPT-2 FINE TUNING TEST:**

Sotto 10/100 epoche i test non sono presenti in quanti i risultati erano scarsi.

1. FILE YAML:

yaml\_config\_template = """

model\_type: llm

base\_model: {model\_path}

input\_features:

  - name: DOMANDE

    type: text

    preprocessing:

      max\_sequence\_length: 300

      truncation: true

      lowercase: true

output\_features:

  - name: RISPOSTE

    type: text

    preprocessing:

      max\_sequence\_length: 300

      truncation: true

    metrics:

      - type: bert\_score

        model: dbmdz/bert-base-italian-uncased

      - type: rouge  # Aggiungi metriche ROUGE per valutare meglio le risposte generate

      - type: perplexity  # Aggiungi perplexity come metrica

prompt:

  template: >-

    Genera una risposta per la seguente domanda, come se fossi un esperto cardiologo.

    Le risposte devono essere empatiche, di senso compiuto e LE FRASI DEVONO TERMINARE CON UN PUNTO e essere lunghe massimo 60 token.

    ### Domanda: {DOMANDE}

    ### Risposta (Deve terminare con un punto e essere di senso compiuto):

generation:

  temperature: 0.6  # Abbassa la temperatura per risposte più coerenti

  max\_new\_tokens: 60

  top\_p: 0.9  # Aggiungi top\_p per migliorare la qualità delle risposte

adapter:

  type: lora

quantization:

  bits: 4

preprocessing:

  split:

    type: fixed

trainer:

  type: finetune

  epochs: 100  # Aumenta il numero di epoche per un fine-tuning più approfondito

  batch\_size: 8  # Aumenta la dimensione del batch se la memoria lo permette

  eval\_batch\_size: 4

  gradient\_accumulation\_steps: 2  # Regola in base alla memoria disponibile

  learning\_rate: 3.0e-5  # Prova con un learning rate più basso

  enable\_gradient\_checkpointing: true

  weight\_decay: 0.01  # Aggiungi weight decay per regolarizzare

  dropout\_rate: 0.1  # Aggiungi dropout per regolarizzare

  learning\_rate\_scheduler:

    decay: cosine

    warmup\_fraction: 0.1  # Mantieni il warmup fraction

    reduce\_on\_plateau: 0

  checkpoint\_interval: 500  # Salva un checkpoint ogni 500 steps

early\_stopping:

  monitor: val\_loss

  patience: 3

  mode: min

logging:

  log\_to\_tensorboard: true

callbacks:

  - type: progress\_bar

  - type: model\_checkpoint

    save\_best\_only: true

    monitor: val\_loss

    mode: min

  - type: early\_stopping

    monitor: val\_loss

    patience: 3

    mode: min

  """

FILE RISULTATI

* Immagine che contiene testo, schermata, Diagramma, Carattere

  Descrizione generata automaticamente

Mostra l'andamento dell'accuratezza dei token (Token Accuracy) durante le epoche di addestramento del tuo modello. L'asse X rappresenta il numero di epoche, mentre l'asse Y rappresenta l'accuratezza dei token.

Cosa sta dicendo il grafico?

\* DECLINO PROGRESSIVO: L'accuratezza dei token inizia a circa 0.06 (6%) e diminuisce progressivamente fino a circa 0.02 (2%) verso la fine del training.

Questo declino indica che il modello sta diventando meno accurato nel prevedere i token corretti man mano che l'addestramento procede.

**>>> Possibile OVERFITTING O PROBLEMI DI ADDESTRAMENTO:**

Un declino continuo e significativo nell'accuratezza dei token può indicare che il modello sta sovraccaricando il rumore presente nel set di dati di training, o potrebbe esserci un problema nella configurazione dell'addestramento.

Un'altra possibilità è che il modello stia incontrando difficoltà con il set di dati e non stia apprendendo in modo efficace.

\*\*SOLUZIONI:\*\*

1. Verifica del Set di Dati: Assicurati che il set di dati di training sia di alta qualità e bilanciato. Dati rumorosi o squilibrati possono influire negativamente sull'addestramento del modello.

2. Regolazione dei Parametri di Addestramento: Prova a regolare l'ipotesi di apprendimento (learning rate), la dimensione del batch, e altre iperparametri per migliorare le prestazioni del modello.

Potresti voler aggiungere tecniche di regolarizzazione come il dropout per prevenire l'overfitting.

3. Early Stopping: Implementa l'early stopping per interrompere l'addestramento se l'accuratezza dei token continua a diminuire, evitando così l'overfitting.

4. Aumento del Set di Dati: Se possibile, aumenta la dimensione del set di dati di training con più esempi pertinenti, che potrebbero aiutare il modello a generalizzare meglio.

5. Analisi dei Modelli di Perdita: Oltre all'accuratezza dei token, osserva anche l'andamento della perdita (loss). Un'analisi combinata delle metriche può fornire ulteriori indicazioni su come migliorare l'addestramento del modello.

* Immagine che contiene testo, schermata, calligrafia, Carattere

  Descrizione generata automaticamente

L'asse X rappresenta il numero di epoche, mentre l'asse Y rappresenta la perplexity del prossimo token. La perplexity misura quanto bene il modello prevede la sequenza di token. Valori più bassi di perplexity indicano una migliore capacità di previsione.

**\* DECLINO PROGRESSIVO:** La perplexity inizia intorno a 28500 e diminuisce progressivamente fino a circa 26000 verso la fine del training. Questo declino indica che il modello sta migliorando nella previsione dei token successivi man mano che l'addestramento procede. **Il modello sta diventando più preciso nelle sue previsioni.**

***Confrontando questo grafico con quello dell'accuratezza dei token che hai fornito in precedenza, notiamo un contrasto interessante:***

**Accuratezza dei Token:** Mostra un declino costante, suggerendo che il modello sta diventando meno preciso nel prevedere i token corretti.

**Next Token Perplexity:** Mostra un declino, suggerendo che il modello sta migliorando nella previsione delle sequenze.

* Immagine che contiene testo, schermata, linea, Diagramma

  Descrizione generata automaticamente

mostra l'andamento della perdita di addestramento (Training Loss) durante le epoche di addestramento del tuo modello. L'asse X rappresenta il numero di epoche, mentre l'asse Y rappresenta il valore della perdita.

**\* DECLINO PROGRESSIVO:** La perdita inizia intorno a 7.5 e diminuisce progressivamente fino a circa 4.0 verso la fine del training. Questo declino indica che il modello sta migliorando nell'addestramento, riducendo l'errore nelle sue previsioni. Indica che il modello sta imparando dai dati di training e sta migliorando nel tempo.

Considera l'aggiunta di tecniche di regolarizzazione come il dropout o la weight decay per prevenire l'overfitting e migliorare la generalizzazione del modello.