yaml\_config\_template = """

model\_type: llm

base\_model: {model\_path}

input\_features:

  - name: DOMANDE

    type: text

    preprocessing:

      truncation: true

      max\_sequence\_length: 140

      lowercase: true

output\_features:

  - name: RISPOSTE

    type: text

    preprocessing:

      max\_sequence\_length: 200

      truncation: true

    metrics:

      - type: bert\_score

        model: dbmdz/bert-base-italian-uncased

      - type: loss

      ### - type: rouge, - type: perplexity

prompt:

  template: >-

    Sei un esperto cardiologo. Rispondi alla seguente domanda in modo empatico e con frasi complete e concise, terminando ogni frase con un punto.

    ### Esempio 1:

    Domanda: Quali sono i sintomi dell'ipertensione?

    Risposta: I sintomi dell'ipertensione includono mal di testa, vertigini e difficoltà respiratorie.

    ### Esempio 2:

    Domanda: Come posso controllare la mia pressione arteriosa?

    Risposta: Puoi controllare la tua pressione arteriosa con una dieta sana, esercizio fisico regolare e farmaci prescritti.

    ### Domanda: {DOMANDE}

    ### Risposta (Assicurati che la tua risposta termini con un punto e abbia senso compiuto):

generation:

  temperature: 0.1    ### per la generazione del testo deve essere molto basso

  max\_new\_tokens: 40

  top\_p: 0.9          ### aiuta a generare testo più coerente e fluido. , il modello avrà un equilibrio tra casualità e coerenza, garantendo che i token generati siano tra quelli con le probabilità più alte.

adapter:

  type: lora

quantization:

  bits: 4

### sessione di validazione

preprocessing:

  split:

    type: random

    probabilities: [0.8, 0.1, 0.1]  ### a gestione di più sottoinsiemi può richiedere risorse aggiuntive.

  global\_max\_sequence\_length: 340

### sessione di validazione

trainer:

  type: finetune

  epochs: 98

  batch\_size: 4

  eval\_batch\_size: 2                       ### indica la dimensione dei batch utilizzata durante la fase di valutazione del modello. (I dati verranno processati in batch di 4 campioni alla volta) >>> ottimizzazione delle risorse.

  learning\_rate: 3.0e-5                     ### definisce la velocità di apprendimento iniziale.

  enable\_gradient\_checkpointing: true

  weight\_decay: 0.01                        ### definisce il tasso di decadimento del peso, che è una tecnica che riduce progressivamente la velocità di apprendimento. (una tecnica che aiuta a prevenire l'overfitting)

  dropout\_rate: 0.1                         ### specifica la probabilità di dropout, che aiuta a prevenire l'overfitting disattivando casualmente unità (neuroni) durante l'allenamento. (10% disattivato)

  learning\_rate\_scheduler:                  ### è la sezione che configura come la velocità di apprendimento viene adattata durante l'allenamento.

    decay: cosine                              ### indica che viene utilizzata una strategia di decadimento coseno. (riduce la velocità di apprendimento seguendo una curva coseno, favorendo un adattamento più dolce verso la fine dell'addestramento)

    warmup\_fraction: 0.1                       ### specifica che la velocità di apprendimento aumenterà gradualmente durante il 10% iniziale dell'addestramento.

  checkpoint\_interval: 5000

### un checkpoint del modello verrà salvato ogni 500 step di addestramento, permettendo il ripristino del modello in caso di interruzioni.

early\_stopping:                             ### monitora la perdita di validazione (val\_loss) e interrompe l'addestramento se la perdita non migliora per 5 epoche consecutive, per prevenire l'overfitting (min: il monitoraggio è configurato per minimizzare la perdita).

  monitor: val\_loss

  patience: 5

  mode: min

logging:                                    ### i log dell'allenamento verranno inviati a TensorBoard.

  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: 5

    mode: min

  """

Immagine che contiene testo, linea, schermata, diagramma

Descrizione generata automaticamente

La **perdita** (loss) è una misura di quanto il modello si discosta dai risultati attesi. Viene calcolata durante l'addestramento del modello e serve come metrica per valutare le sue prestazioni. La **perdita di validazione** viene calcolata allo stesso modo, ma utilizzando un set di dati separato (dati di validazione) che non viene utilizzato per l'addestramento, al fine di misurare come il modello generalizza su dati non visti.

L'asse delle X nel grafico rappresenta le **epoche**. Un'epoca è un singolo passaggio attraverso l'intero set di dati di addestramento. Ogni epoca permette al modello di aggiornare i suoi pesi sulla base degli errori commessi.

L'asse delle Y rappresenta il valore della **perdita**. La perdita viene calcolata utilizzando una funzione di perdita specifica, che può variare a seconda del tipo di problema.

Immagine che contiene testo, schermata, linea, Diagramma

Descrizione generata automaticamente

Mostra quanto il modello sta generalizzando su dati non visti. Una riduzione costante di questa perdita indica che il modello sta generalizzando bene. Se la perdita di validazione inizia ad aumentare mentre la perdita di addestramento continua a diminuire, indica overfitting.

Immagine che contiene schermata, testo, Diagramma, linea

Descrizione generata automaticamente

**Riduzione della perdita:** Sia la perdita di addestramento che quella di validazione diminuiscono nel tempo, il che indica che il modello sta imparando e migliorando le sue previsioni. Le perdite iniziano con un valore elevato e diminuiscono significativamente durante le prime epoche.

**Convergenza:** Verso le ultime epoche, i valori di perdita sia per l'addestramento che per la validazione tendono a convergere e a stabilizzarsi, suggerendo che il modello ha raggiunto un punto in cui l'ulteriore apprendimento significativo è minimo.

**Perdita di convalida:** la perdita di convalida è costantemente inferiore a quella di addestramento, il che è un buon segno perché indica che il modello non si adatta eccessivamente ai dati di addestramento e si generalizza bene ai dati non visti.

Immagine che contiene testo, schermata, Diagramma, diagramma

Descrizione generata automaticamente

La perplessità (Perplexity) è una misura di quanto un modello di linguaggio è confuso dalle sequenze che deve prevedere. In generale, una perplessità più bassa indica un modello migliore.

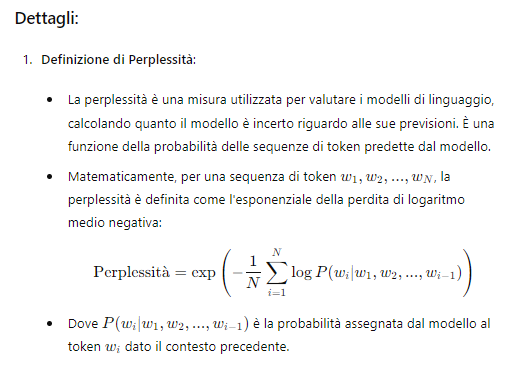
Asse delle X (Epochs):

Rappresenta il numero di epoche, cioè il numero di volte che il modello ha visto l'intero set di dati di addestramento. Nel grafico, vediamo un totale di 100 epoche.

Asse delle Y (Next Token Perplexity):

Indica la perplessità per il token successivo. Una perplessità più bassa significa che il modello è più sicuro delle sue predizioni.

* Inizio: All'inizio dell'addestramento (prime epoche), la perplessità è molto alta (circa 27000), indicando che il modello è molto confuso.
* Progressione: Man mano che l'addestramento procede, la perplessità diminuisce rapidamente nelle prime epoche, mostrando che il modello sta imparando a fare previsioni migliori.
* Convergenza: Dopo circa 20-30 epoche, la riduzione della perplessità diventa più lenta, ma continua a diminuire fino a stabilizzarsi intorno a un valore di circa 20000 verso la fine dell'addestramento.



Questa non è misurata in unità fisiche come metri o secondi, ma è un valore numerico puro.

Immagine che contiene testo, schermata, schermo, Diagramma

Descrizione generata automaticamente

Il grafico rappresenta la "Token Accuracy" (accuratezza dei token) durante il processo di addestramento del modello di linguaggio.

Asse Y

Token Accuracy: L'asse Y rappresenta l'accuratezza dei token. L'accuratezza dei token misura la percentuale di token (parole o caratteri) predetti correttamente dal modello rispetto al totale dei token predetti.

**Percentuale**: L'unità di misura sull'asse Y è una percentuale, ma in questo caso è rappresentata come un valore frazionario (decimale). Ad esempio, un valore di 0.020 corrisponde al 2% di accuratezza, un valore di 0.010 corrisponde all'1% di accuratezza, e così via.

* Inizio dell'Addestramento: All'inizio dell'addestramento, l'accuratezza dei token è molto bassa, indicando che il modello è inizialmente molto impreciso nelle sue previsioni.
* Rapido Miglioramento Iniziale: Nei primi cicli di addestramento, l'accuratezza dei token aumenta rapidamente, mostrando che il modello sta imparando e migliorando.
* Stabilizzazione: Dopo circa 20 epoche, l'accuratezza si stabilizza intorno a un valore basso, che resta costante fino alla fine dell'addestramento.