Relazione – Settimana 2

Gruppo di Lavoro

1 aprile 2025

Introduzione e Obiettivi

Nel corso della seconda settimana di lavoro, ci siamo concentrati sulla definizione del **dataset iniziale** necessario all'addestramento di BERT, sull'analisi del flusso di elaborazione e sulla scelta degli strumenti di traduzione per gestire risposte in lingue diverse dall'inglese. In particolare, abbiamo approfondito la *strategia di test* che prevede di utilizzare una versione più leggera di BERT su Google Colab, per poi passare a una variante più potente da eseguire sul cluster del dipartimento.

Definizione del Dataset

Dopo una valutazione delle risorse disponibili e in accordo con il professore, abbiamo deciso di creare un **pool iniziale di 4.000 esempi**, di cui:

- **2.000 esempi di risposte "non jailbreak"**, ritenute valide o in linea con le policy;
- **2.000 esempi di risposte "jailbreaked"**, in cui l'LLM è stato indotto a produrre output non conformi o a violare determinati vincoli.

La soglia di 2.000 esempi per parte è stata scelta sia per la disponibilità limitata di risposte jailbreaked, sia per mantenere la fase di *fine-tuning* più gestibile dal punto di vista computazionale. Le nostre ricerche iniziali suggerivano un range compreso tra 2.000 e 5.000 esempi per parte, e la scelta di rimanere su 2.000 è un compromesso tra *qualità* del dataset e *fattibilità* dell'esperimento.

Scrematura del Dataset Abbiamo previsto una fase di *scrematura* (prevista per la prossima settimana) non appena il professore ci metterà a disposizione il *dataset iniziale* generato durante i test di *jailbreaking*. Questa scrematura servirà a eliminare dati duplicati, inconsistenti o privi di etichette affidabili, così da garantire la massima coerenza del set di addestramento.

Strategia di Test su Google Colab

Uno dei punti centrali emersi questa settimana riguarda la **necessità di testare il processo di fine-tuning** su un modello più leggero di BERT, da eseguire su *Google Colab*. Questa scelta nasce dall'esigenza di:

- Validare rapidamente l'approccio su un dataset di dimensioni ridotte, monitorando metriche come accuratezza, precisione e recall;
- Familiarizzare con il *fine-tuning* di BERT in ambiente Colab, identificando eventuali colli di bottiglia (tempo di training, memoria GPU, ecc.);
- Evitare sprechi di risorse sul cluster del dipartimento, riservandolo a fasi successive, più impegnative computazionalmente.

Una volta completata questa fase di test e **validata** l'efficacia del dataset (in termini di risultati ottenuti), prevediamo di:

1. Eseguire il *fine-tuning* di una *versione più potente di BERT* sul cluster del dipartimento, sfruttando l'hardware più performante;
2. Salvare il modello addestrato su *Hugging Face*, in modo da poterlo integrare facilmente in diversi ambienti di sviluppo *Python*.

Introduzione di un Layer di Traduzione

Rianalizzando lo *schema generale* del sistema, ci siamo resi conto che non si può assumere che le risposte dell'LLM siano esclusivamente in lingua inglese. Per questo motivo, abbiamo deciso di **introdurre un layer di traduzione** che converta eventuali testi in lingue diverse dall'inglese, in modo da standardizzare l'input prima di inviarlo a BERT.

Librerie Python per la Traduzione Abbiamo esplorato diverse *librerie Python* che consentono di effettuare traduzioni in modo *batch* o *on-the-fly*. Alcune di queste librerie si interfacciano direttamente con servizi esterni (ad esempio *Google Translate*), mentre altre si basano su modelli open source. L'idea è di scegliere la soluzione che offra il miglior compromesso tra:

- *Qualità della traduzione*: importante per non introdurre distorsioni semantiche nel testo di input;
- *Velocità di esecuzione*: necessaria per non rallentare eccessivamente il *pipeline* di valutazione;
- *Costi e limiti di API*: in caso di utilizzo di servizi esterni.

Schema Aggiornato della Soluzione

Nella Figura 1 riportiamo il **nuovo schema** del sistema, in cui si evidenzia il *layer di traduzione* prima dell'elaborazione da parte di BERT.

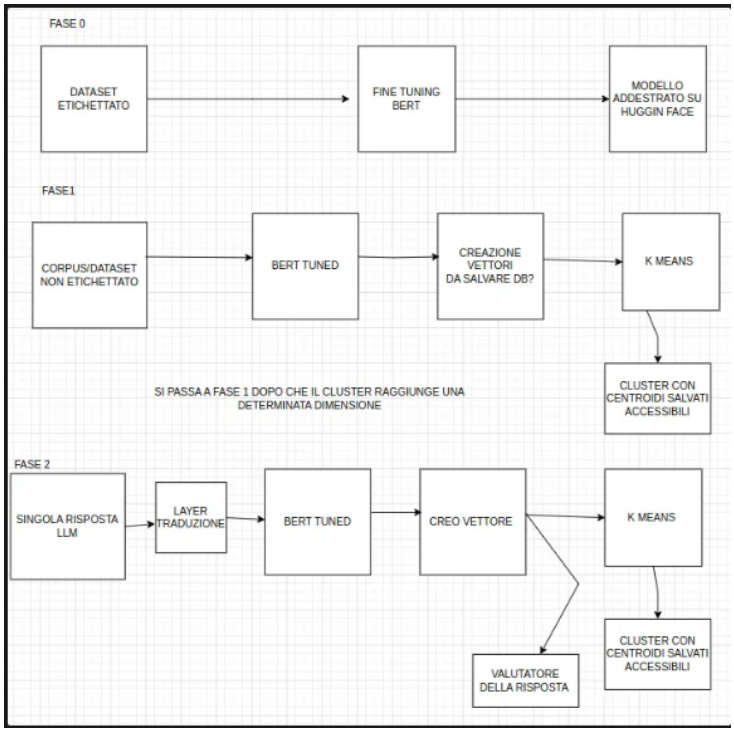

Figura 1: Schema aggiornato della soluzione, con l'aggiunta del layer di traduzione.

Analisi dei Dati per il Clustering

Oltre all'addestramento di BERT, ci siamo posti il problema di quante risposte saranno necessarie per la fase di *clustering* tramite *k-means*. In particolare:

- **Dataset per il clustering**: potremmo utilizzare lo stesso *dataset* di addestramento di BERT, integrato con ulteriori esempi, per ottenere una varietà di vettori sufficientemente ampia da garantire la robustezza dei centroidi.
- **Aggiornamento dinamico o statico**: non abbiamo ancora stabilito se, a regime, i cluster dovranno aggiornarsi con l'arrivo di nuove risposte. Questa scelta dipenderà dalla *qualità* e *stabilità* delle previsioni di BERT e dal *tasso* con cui si generano nuovi dati.

Ricerca Implementativa e Documentazione

Abbiamo proseguito il lavoro di ricerca sulle **soluzioni implementative in Python**, analizzando vari *video* e *materiali* relativi al *fine-tuning* di BERT. Come di consueto, tutto il materiale raccolto (librerie, tutorial, esempi di codice) sarà inserito o collegato su *Notion*, così da garantire la condivisione del sapere e la possibilità di riprodurre i passaggi di configurazione in futuro.

Conclusioni e Prospettive

In sintesi, la seconda settimana di lavoro si è concentrata su:

- **Definire** la dimensione e la composizione del dataset iniziale (4.000 esempi bilanciati tra risposte jailbreaked e non jailbreaked);
- **Pianificare** la fase di test su *Google Colab* con una versione leggera di BERT, per poi passare a un modello più potente sul cluster del dipartimento;
- **Inserire** un layer di traduzione nel *pipeline*, al fine di gestire risposte multilingue e garantire la coerenza dell'input a BERT;
- **Analizzare** le esigenze in termini di dati per la creazione dei cluster *k-means*, valutando la possibilità di un aggiornamento dinamico.

Nella prossima settimana, contiamo di:

- Ricevere e *scremare* il dataset dal professore, eliminando dati non validi o ridondanti;
- Avviare i primi esperimenti di *fine-tuning* su BERT Base in ambiente Colab;
- Proseguire lo studio dei *modelli di traduzione* e definire la pipeline definitiva di elaborazione del testo.