UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Utilizzo di Large Language Models e Fine-Tuning per Supportare il Processo di Peer Review

RELATORE

Prof. Fabio Palomba

Dott. Stefano Lambiase

Università degli Studi di Salerno

CANDIDATO

Antonio D'Auria

Matricola: 0512114149

Anno Accademico 2023-2024

Abstract

Il processo di Peer Review è essenziale per garantire la qualità della ricerca scientifica, ma affronta crescenti difficoltà a causa dell'aumento degli articoli sottomessi e della limitata disponibilità di revisori qualificati. Questo lavoro esplora l'utilizzo di modelli di Deep Learning per assistere i revisori, generando revisioni preliminari che supportino il processo valutativo. Attraverso il Fine-Tuning di diversi LLM su dataset di articoli scientifici e relative revisioni, è stato possibile adattare i modelli per gestire tali documenti. I risultati dimostrano l'efficacia di questi modelli non solo nella generazione di contenuti coerenti e rilevanti, ma anche nella capacità di fornire analisi critiche, evidenziando come siano in grado di catturare le strutture dei testi scientifici e di offrire valutazioni utili ai revisori. Nonostante i progressi ottenuti, restano sfide significative legate alla gestione di documenti particolarmente lunghi, dove mantenere un ampio contesto risulta cruciale, e alla comprensione dei contenuti complessi degli articoli, spesso basati su riferimenti allo stato dell'arte. L'adozione di modelli e tecniche ancora più avanzate, insieme alla possibilità di accedere a risorse computazionali adeguate, rappresentano direzioni promettenti per future ricerche. Questo lavoro pone solide basi per lo sviluppo di strumenti in grado di assistere i revisori, contribuendo a rendere il processo di revisione scientifica più rapido ed efficiente.

Indice

Elenco delle figure

CAPITOLO 1

Introduzione

1.1 Contesto e limitazioni

Il processo di Peer Review rappresenta una componente fondamentale nella diffusione della conoscenza scientifica, garantendo l'integrità, la qualità e l'affidabilità dei risultati pubblicati [2]. Attraverso le Peer Review, i ricercatori valutano criticamente i lavori dei propri colleghi, contribuendo a migliorare la qualità della ricerca accademica e ad assicurare che i risultati diffusi rispondano agli standard metodologici e scientifici. Questo lavoro richiede competenze avanzate e un impegno considerevole in termini di tempo, senza che vi sia, nella maggior parte dei casi, una compensazione economica per i revisori [3]. La revisione scientifica è, infatti, un'attività prevalentemente volontaria, resa possibile dalla dedizione dei ricercatori verso la comunità accademica. Data la crescente quantità di articoli scientifici sottomessi ogni anno e la limitata disponibilità di revisori qualificati, diventa sempre più complesso assicurare che tutti i manoscritti siano valutati con la dovuta attenzione e tempestività [4]. In questo contesto, disporre di uno strumento automatizzato che supporti o addirittura generi revisioni critiche preliminari rappresenterebbe un'enorme risorsa per il sistema scientifico. Uno strumento di questo tipo potrebbe non solo alleviare il carico di lavoro dei ricercatori, ma anche ridurre i tempi di valutazione e migliorare

l'accessibilità a recensioni di alta qualità.

In quest'ottica, l'impiego di modelli di linguaggio di grandi dimensioni offre una promettente soluzione per supportare e automatizzare il processo di revisione scientifica. Gli LLM, addestrati su grandi quantità di dati e ottimizzati per comprendere e generare linguaggio naturale in modo contestuale, possono essere in grado di analizzare un paper e fornire una valutazione critica e strutturata [5]. Questi modelli, infatti, possono essere calibrati per generare una struttura coerente e fornire un'analisi critica del contenuto, replicando in parte il linguaggio e l'approccio tipici di una revisione accademica [6].

Nonostante le potenzialità offerte dagli LLM, automatizzare il processo di Peer Review rimane una sfida complessa. Innanzitutto, i paper e le revisioni scientifiche non seguono una struttura fissa: il formato varia ampiamente a seconda della rivista, della disciplina e di specifici requisiti editoriali, rendendo difficile per un modello di linguaggio adattarsi a una struttura univoca [7]. Inoltre, il contenuto di cui questi modelli devono occuparsi è di notevole lunghezza; i paper scientifici contengono spesso decine di pagine, mentre anche le revisioni più concise possono estendersi su più paragrafi e richiedere un'analisi dettagliata. Questa elevata quantità di testo rappresenta un ulteriore ostacolo per gli LLM, che devono essere in grado di gestire sequenze lunghe senza perdere coerenza e rilevanza [8]. Infine, sia i paper che le revisioni includono contenuti tecnici e accademici, spesso allineati allo stato dell'arte della ricerca, richiedendo quindi una comprensione profonda e precisa da parte del modello. Di conseguenza, replicare l'accuratezza e la competenza di un revisore umano rappresenta una sfida significativa per gli LLM, i quali devono ancora affrontare limiti inerenti alla comprensione semantica e al contesto specialistico [9].

1.2 Obiettivo del lavoro

L'obiettivo di questo lavoro è progettare e sviluppare un sistema basato su modelli di linguaggio di grandi dimensioni (LLM) per la generazione automatica di revisioni scientifiche. Il sistema mira a produrre revisioni accurate, coerenti e formali, utilizzando articoli scientifici come input. Questo approccio potrebbe rappresentare un'importante innovazione nel processo di Peer Review, supportando i ricercatori

nella valutazione dei manoscritti e contribuendo a ridurre i tempi e il carico di lavoro associati.

Per raggiungere tale scopo, sono stati selezionati e ottimizzati diversi LLM tramite tecniche di Fine-Tuning, adattandoli al task specifico di generazione di revisioni critiche. I modelli adottati, tra cui *T5 Base*, *Flan-T5 XXL* e *LED Large*, sono stati valutati per confrontarne le prestazioni su parametri fondamentali come la coerenza, l'accuratezza e la rilevanza delle revisioni generate.

Il lavoro è stato condotto nel pieno rispetto dei principi di integrità etica e delle policy accademiche sull'uso di GenAI (2.5), rappresentando una simulazione realistica del processo di Peer Review, effettuata su artefatti interni al laboratorio di ricerca con cui è stata svolta la tesi e basata su dati prodotti prima dell'avvento delle tecnologie di Intelligenza Artificiale Generativa.

L'obiettivo finale è stato quello di identificare il modello e le configurazioni più efficaci, tenendo conto dei limiti dei modelli, dei vincoli imposti dalle risorse computazionali e dalle caratteristiche del dataset utilizzato. In particolare, il lavoro si propone di dimostrare come le tecnologie basate su LLM possano essere applicate al contesto accademico, offrendo uno strumento innovativo per migliorare l'efficienza e la qualità del processo di Peer Review.

1.3 Contributi del lavoro

Il lavoro svolto presenta diversi contributi rilevanti nel contesto della generazione automatica di revisioni scientifiche basata su LLM. I principali contributi sono stati:

- Training, valutazione e confronto di vari LLM: Sono stati utilizzati diversi LLM per effettuare la generazione automatica di revisioni scientifiche. Attraverso l'impiego di tecniche di Fine-Tuning, i modelli *T5 Base*, *Flan-T5 XXL* e *LED Large*, sono stati addestrati, valutati e infine confrontati per identificare la configurazione più adatta al task specifico, sperimentando diverse configurazioni dei parametri per ciascun modello. Il sistema è stato progettato per produrre revisioni critiche accurate, coerenti e formali, considerando i limiti imposti dal contesto (1.1).

- Valutazione comparativa dei modelli: Sono stati esaminati e confrontati i risultati di diversi modelli, analizzandone le prestazioni in termini di coerenza, accuratezza e formalità delle revisioni generate. Questa analisi ha consentito di identificare i punti di forza e i limiti di ciascun modello, fornendo indicazioni utili per l'applicazione di LLM in contesti accademici.

- Ottimizzazione delle risorse computazionali: L'adozione di tecniche come la Quantizzazione e l'adattamento di LoRA ha permesso di gestire efficacemente modelli di grandi dimensioni come *Flan-T5 XXL*, riducendo il consumo di memoria e migliorando l'efficienza del training. Sono state inoltre configurati parametri adattativi di training per bilanciare la qualità dei risultati con i limiti imposti dall'hardware disponibile.

In sintesi, il lavoro mira a mostrare l'integrazione degli LLM nel contesto accademico, al fine di fornire strumenti e metodologie utili per migliorare il processo di generazione di revisioni scientifiche e per esplorare ulteriormente l'applicazione di modelli di linguaggio avanzati in ambiti complessi e strutturati.

1.4 Struttura della tesi

La tesi si articola in quattro capitoli principali, ciascuno dei quali affronta un aspetto specifico del lavoro svolto, dalle basi teoriche all'approccio risolutivo proposto, fino alla sintesi finale dei risultati ottenuti.

- Introduzione: Viene presentato il contesto generale del lavoro, mettendo in evidenza le sfide e le limitazioni legate allo sviluppo di una soluzione per la generazione automatica di revisioni scientifiche. Si delineano l'obiettivo del lavoro e i principali contributi offerti, concludendo con una panoramica della struttura complessiva della tesi.

- Background e stato dell'arte: Questo capitolo analizza i concetti fondamentali del processo di Peer Review, del Machine Learning e del Deep Learning, con particolare attenzione al Natural Language Processing (NLP) e ai modelli di

linguaggio di grandi dimensioni (LLM). Viene inoltre approfondito il ruolo dell'Intelligenza Artificiale Generativa (GenAI) nel contesto accademico, mettendo in evidenza le sfide etiche e tecniche legate all'adozione di queste tecnologie.

- Proposta: Viene descritto in dettaglio il metodo sviluppato per la generazione automatica di revisioni scientifiche. Il capitolo include la descrizione dei modelli selezionati, le configurazioni tecniche utilizzate per il training e l'inferenza, le tecniche di ottimizzazione adottate per superare i vincoli computazionali e le metriche impiegate per la valutazione delle prestazioni. Vengono inoltre discusse le limitazioni incontrate durante lo sviluppo e le soluzioni implementate per affrontarle.

- Conclusioni: Questo capitolo sintetizza i risultati principali del lavoro, evidenziando i contributi ottenuti. Vengono discusse le implicazioni pratiche dell'utilizzo di LLM per la generazione di revisioni scientifiche e vengono proposte possibili direzioni future per migliorare e ampliare l'applicazione di LLM nel contesto accademico.

CAPITOLO 2

Background e stato dell'arte

2.1 Il Processo di Peer Review

Il processo di Peer Review è definito come un processo in cui il lavoro accademico, la ricerca o le idee di un autore vengono sottoposte al vaglio di altri ricercatori esperti dello stesso settore [10].

Lo scopo delle Peer Review è quello di garantire che gli autori rispettino gli standard elevati dell'effettuare una pubblicazione scientifica e di controllare la diffusione dei dati di ricerca, garantendo che affermazioni non fondate, interpretazioni inaccettabili o opinioni personali non vengano pubblicate senza una revisione esperta preliminare. Si tratta quindi di valutare la validità e la qualità degli articoli destinati alla pubblicazione, al fine di mantenerne l'integrità metodica ed evitare la diffusione di idee invalide o di scarsa qualità [2].

Dal punto di vista editoriale, le Peer Review possono essere pensate come un "filtro" per i contenuti, il cui obiettivo è quello di indirizzare gli articoli di alta qualità verso riviste sempre più considerevoli, aggiungendo quindi ad essi un valore maggiore.

Il sistema di Peer Review non è esente da critiche. Gli studi di J Ray et al. [11] dimostrano come su un totale di 350 paper rifiutati, 240 sono stati pubblicati altrove

dopo una media di 552 giorni; di 226 articoli di ricerca respinti, 159 (70%) sono stati pubblicati in riviste specialistiche. W. G. Baxt et al. [12], invece, si pongono la seguente domanda: *"Who reviews the reviewers?" - Chi valuta i revisori?*. La loro ricerca dimostra che un articolo è stato inviato a 262 revisori, di cui 203 (78%) hanno risposto. Tra i 199 revisori che hanno fornito una raccomandazione, 15 hanno suggerito l'accettazione, 117 il rifiuto, e 67 la revisione. I revisori che hanno raccomandato il rifiuto hanno identificato una percentuale significativamente maggiore di errori principali (39, 1%) e minori (25, 2%) rispetto a quelli che hanno suggerito accettazione o revisione. Inoltre, il 68% dei revisori non ha rilevato che le conclusioni non erano supportate dai risultati. Dunque il processo di Peer Review pur essendo ampiamente considerato valido e fondamentale per garantire la qualità della ricerca scientifica, presenta ancora delle criticità che richiedono miglioramenti.

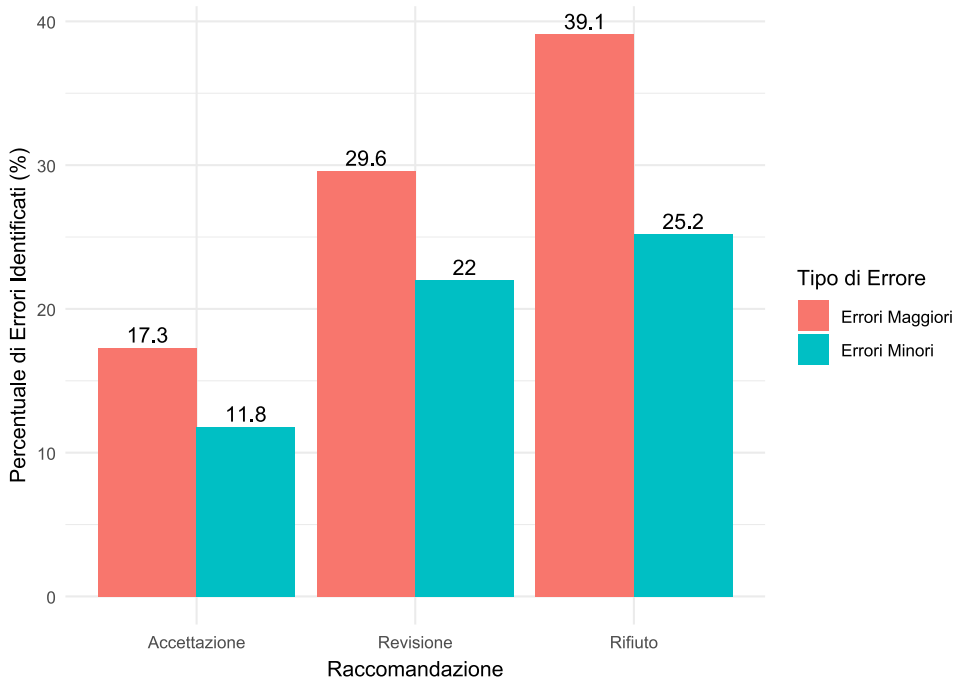

Figura 2.1: Percentuale di errori identificati dai revisori in base alla raccomandazione fornita

2.1.1 Processo di pubblicazione e sottomissione di articoli

Il processo di Peer Review può essere ampiamente riassunto in 10 fasi [13], anche se queste possono variare leggermente tra le diverse riviste.

1. Sottomissione dell'articolo: l'autore sottomette l'articolo alla rivista, solitamente attraverso un sistema online come ScholarOne Manuscripts. Occasionalmente, delle riviste permettono anche l'invio tramite e-mail.

2. Valutazione dell'ufficio editoriale: l'ufficio editoriale verifica che l'articolo rispetti le linee guida per gli autori della rivista. Durante questa fase, la qualità non viene ancora valutata.

3. Valutazione del caporedattore (EIC): il caporedattore valuta l'articolo, considerando il suo scopo, l'originalità e i meriti. A questo punto potrebbe decidere di rifiutare l'articolo.

4. Assegnazione a un editore associato (AE): alcune riviste presentano un editore associato che gestisce il processo di Peer Review. Se presente, viene coinvolto in questa fase.

5. Invito ai revisori: l'editore invita individualmente i revisori che ritiene opportuni. Se necessario, vengono mandati ulteriori inviti, fino al raggiungimento del numero richiesto di revisori (solitamente 2 o 3, a seconda delle linee guida della rivista).

6. Risposta agli inviti: i revisori valutano l'invito in base alle loro competenze, conflitti di interesse e disponibilità. Procedono con l'accettare o col rifiutare l'invito.

7. Conduzione della revisione: il revisore legge l'articolo più volte. Una prima lettura viene data per poter avere un impressione iniziale del lavoro. Se sorgono problemi gravi, il ricercatore potrebbe voler rifiutare l'articolo sin da subito. Altrimenti, procederà leggendo l'articolo più volte, creando un analisi dettagliata punto per punto. La revisione viene successivamente pubblicata sulla rivista, accompagnata dalla raccomandazione del revisore.

8. Valutazione delle revisioni da parte della rivista: l'editore considera tutte le revisioni ricevute per poi fornire l'esito della sua decisione. Se ci sono differenze significative tra le revisioni, può invitare un revisore aggiuntivo per una nuova opinione.

9. Comunicazione della decisione: l'editore invia un'e-mail all'autore includendo ogni commento rilevante.

10. Passi successivi: se accettato, l'articolo va in produzione. Se rifiutato o rimandato per revisione, vengono forniti commenti costruttivi per migliorarlo.

2.1.2 Scrittura di una Peer Review

Scrivere una Peer Review non è solo una questione di valutare il merito scientifico di un articolo, ma richiede un'analisi approfondita e imparziale che contribuisca al miglioramento della qualità della ricerca pubblicata. Il revisore deve esaminare attentamente la chiarezza del testo, la validità della metodologia, la rilevanza dei risultati e la coerenza delle conclusioni rispetto alle prove fornite. Non esiste un metodo generale o universale per effettuare una Peer Review, poiché i requisiti possono variare a seconda della rivista e del campo di ricerca. Tuttavia, nel corso degli anni, le comunità scientifiche hanno sviluppato diversi formati standard per garantire una qualità elevata e rispettare le esigenze specifiche del settore di riferimento, e.g., lo standard CONSORT [14] è stato creato per aiutare gli autori a presentare gli Studi Clinici Randomizzati (RCTs) in modo chiaro, trasparente e completo, mentre, lo standard PRISMA [15] è utilizzato principalmente per le revisioni sistematiche e la meta-analisi. Il revisore deve quindi essere consapevole di tali standard e assicurarsi che l'articolo li rispetti, contribuendo così a migliorare la chiarezza e l'integrità del manoscritto.

Sebbene non esista un metodo universalmente accettato, vi è comunque un insieme di buone pratiche e linee guida che i revisori dovrebbero seguire, indipendentemente dallo standard considerato. Queste pratiche includono la lettura approfondita del manoscritto per identificare eventuali problemi metodologici, errori logici o ambiguità nel testo, come suggerito nelle linee guida per i revisori, fornite sulla piat-

taforma di Wiley Author Services [16]. Quando si esegue una Peer Review, il processo può essere suddiviso in diverse fasi atte a garantire una valutazione approfondita e costruttiva.

1. Prima lettura: la prima lettura, oltre che fornire una panoramica generale dell'articolo, serve a identificare eventuali problematiche evidenti. Se emergono difetti significativi, come metodologie inadeguate o una scarsa pertinenza, si può decidere di porre immediato rifiuto senza ulteriori analisi.

2. Seconda lettura: se l'articolo supera la prima valutazione, si procede con una lettura più dettagliata. Questa fase richiede un'analisi approfondita di tutte le sezioni chiave del lavoro, ponendo molta attenzione sulla metodologia, i risultati e la discussione.

3. Strutturare la revisione: la revisione deve essere ben strutturata e organizzata in modo chiaro, suddividendo i commenti per le varie sezioni in maniera tale da fornire un feedback mirato.

4. Stile e presentazione: è fondamentale mantenere un tono professionale e costruttivo durante tutto il processo. La critica dovrebbe essere focalizzata sul miglioramento del lavoro, evitando commenti personali o giudizi non costruttivi.

Seguendo un approccio strutturato, con attenzione ai dettagli e un feedback costruttivo, i revisori possono quindi svolgere un ruolo cruciale nel processo di Peer Review, promuovendo l'integrità e la solidità della letteratura scientifica.

2.1.3 Tipi di Peer Review

Il processo di Peer Review ha subito molte trasformazioni nel corso del tempo, con l'aumento della produzione scientifica e la necessità di garantire la qualità delle pubblicazioni, si è evoluto in vari modelli che differiscono principalmente per il grado di anonimato tra autori e revisori [17]. I tre principali tipi di Peer Review sono il *Single-Anonymized*, il *Double-Anonymized* e l'*Open Peer Review*. Successivamente sono stati sviluppati nuovi modelli come il *Triple-Anonymized*, il *Transparent*, il *Collaborative* e il

Post Pubblication, questi ultimi rappresentano variazioni chiave rispetto all'approccio standard. Il sistema di revisione è in costante sviluppo, con continui esperimenti su nuovi modelli e adattamenti di quelli già esistenti.

Tipo di Peer Review	**Descrizione**
Single Anonymized	L'identità del revisore non è visibile all'autore, l'identità dell'autore è visibile al revisore, l'identità di revisore e autore è visibile all'editore.
Double Anonymized	L'identità del revisore non è visibile all'autore, l'identità dell'autore non è visibile al revisore, l'identità di revisore e autore è visibile all'editore.
Triple Anonymized	L'identità del revisore non è visibile all'autore, l'identità dell'autore non è visibile al revisore, l'identità di revisore e autore non è visibile all'editore.
Open Peer Review	L'identità del revisore è visibile all'autore, l'identità dell'autore è visibile al revisore, l'identità di revisore e autore è visibile all'editore.
Transparent Peer Review	Il report di revisione è pubblicato insieme all'articolo. Il revisore può scegliere se condividere la propria identità.
Collaborative	Due o più revisori lavorano insieme per fornire un report unificato. Oppure, l'autore revisiona il manoscritto sotto la supervisione di uno o più revisori.
Post Publication	La revisione, sollecitata o non sollecitata, avviene su un articolo già pubblicato. Non esclude altre forme di peer review.

Tabella 2.1: Tipi di Peer Review e le loro caratteristiche

Ogni modello ha i propri vantaggi e limitazioni, e la scelta del metodo più appropriato dipende dalle esigenze specifiche della comunità scientifica e della rivista. È chiaro che l'obiettivo comune rimane quello di garantire qualità e imparzialità.

2.2 Machine Learning e Deep Learning

L'Intelligenza Artificiale (IA), in particolare il Machine Learning (ML), ha subito una crescita esponenziale negli ultimi anni diventando sempre più utilizzata in ambiti come la classificazione, il calcolo avanzato, la previsione di dati futuri e l'elaborazione del linguaggio naturale. Il Machine Learning è un sottoinieme dell'Intelligenza Artificiale che fornisce ai sistemi la capacità di apprendere ed evolversi automaticamente dall'esperienza, senza essere esplicitamente programmati [18]. Il concetto di autoapprendimento nell'intelligenza artificiale si riferisce alla capacità dei computer di apprendere dai dati, in modo analogo a come gli esseri umani acquisiscono conoscenze dall'esperienza. Le macchine non necessitano di essere programmate per affrontare ogni singola situazione; piuttosto, attraverso l'osservazione di numerosi esempi, sono in grado di dedurre autonomamente come risolvere problemi analoghi in futuro.

2.2.1 Tipi di dati

L'analisi dei dati rappresenta uno degli aspetti fondamentali in ogni sistema di Machine Learning o Intelligenza Artificiale. Prima che i modelli possano imparare e fare previsioni, i dati devono essere acquisiti, pre-processati e organizzati in un formato che possa essere facilmente interpretato dalle macchine [19]. In questo contesto, i dati possono essere classificati in tre cateogorie principali:

- Dati strutturati: si tratta di dati che presentano una struttura rigida, sono tipicamente organizzati in tabelle con righe e colonne, come i fogli di calcolo o i database relazionali. Ciascun elemento presenta una posizione specifica e un significato definito, il che rende tali dati facili da gestire e analizzare.

- Dati semi-strutturati: questi dati non seguono uno schema tabellare rigido, purtuttavia presentano una struttura interna. Esempi tipici sono documenti JSON o XML, che contengono tag o campi predefiniti per facilitare l'organizzazione delle informazioni. Possono essere parzialmente organizzati per fornire ai modelli di IA un'analisi più agevole.

- Dati non strutturati: questa è la categoria più ampia, comprende dati come testi, immagini, foto, audio e video. Non è presente una struttura organizzata, per tale motivo l'elaborazione di dati non strutturati ha portato alla necessità di sviluppare tecniche avanzate di estrazione delle informazioni.

La scelta del tipo di dati da utilizzare dipende dal compito specifico che si intende affrontare. Generalmente i dati strutturati sono ideali per compiti che presentano una natura di per sé già organizzata, come nelle applicazioni finanziarie o gestionali. I dati semi-strutturati, invece, offrono flessibilità per scenari in cui alcune informazioni sono organizzate ma richiedono ancora una certa elaborazione. Infine, i dati non strutturati, come testo, immagini o video, sono fondamentali per compiti di elaborazione avanzata, come il riconoscimento del linguaggio naturale o l'analisi delle immagini.

2.2.2 Tipi di apprendimento

Nel Machine Learning i modelli devono imparare a riconoscere pattern e a prendere decisioni in base ai dati forniti. Tuttavia, il modo in cui i modelli apprendono può variare a seconda della struttura e della disponibilità dei dati, nonché del compito specifico da svolgere. L'apprendimento è solitamente suddiviso in tre tipi principali: l'approccio *Predittivo* o *Supervisionato*, l'approccio *Descrittivo* o *Non Supervisionato* e l'approccio *Per Rinforzo*. Per illustrare le differenze tra i vari approcci, utilizzerò la spiegazione fornita da *Kevin P. Murphy* nel suo libro *Machine Learning: A Probabilistic Perspective* [20].

Apprendimento Supervisionato In questo approccio l'obiettivo è imparare una mappatura dagli input x agli output y, dato un insieme di coppie input-output etichettate $D = \{(x_i, y_i)\}_{i=1}^{N}$. Qui, D è chiamato *Training Set* e N è il numero di esempi di addestramento. Si consideri ogni input di addestramento x_i come un vettore di numeri a D dimensioni, che rappresentano, per esempio, l'altezza e il peso di una persona. Questi sono chiamati *Feature*, *Attributi* o *Covariate*. In generale, x_i potrebbe essere un oggetto strutturato complesso come un'immagine, una frase, una serie temporale, etc. Analogamente anche l'output y_i può essere in linea di principio qualsiasi cosa, ma di solito si assume che sia una variabile

categorica nominale proveniente da un insieme finito. Quando y_i è categorico, il problema è noto come *Classificazione* o *Riconoscimento di Pattern*, quando y_i assume valori reali, il problema è noto come *Regressione*. Un'altra variante nota come *Ordinal Regression*, si verifica quando lo spazio delle etichette Y ha un ordinamento naturale.

Apprendimento Non Supervisionato Nell'approccio Non Supervisionato vengono forniti soltanto gli input $D = \{x_i\}_{i=1}^{N}$ e l'obiettivo è trovare pattern interessanti nei dati. Si tratta di un problema molto meno definito poiché non viene specificato quale tipo di pattern cercare siccome i dati non sono etichettati. Una tecnica comune che fa utilizzo di questo tipo di apprendimento è il *Clustering* dove il modello raggruppa i dati in categorie o cluster basati su somiglianze. Algoritmi comuni includono:

- K-means: divide i dati in K gruppi, cercando di minimizzare la distanza tra i punti e il centroide del gruppo.
- DBSCAN: identifica cluster di densità, individuando punti che formano aree densamente popolate.

Apprendimento Per Rinforzo Questo tipo di apprendimento è utile per insegnare al modello come agire o comportarsi quando vengono forniti segnali occasionali di ricompensa o punizione. L'obiettivo è massimizzare una ricompensa cumulativa nel tempo, scegliendo le azioni che portano ai risultati migliori.

I diversi tipi di apprendimento, offrono quindi strumenti flessibili per affrontare una vasta gamma di problemi di Machine Learning, ciascuno con le proprie caratteristiche e applicazioni ottimali.

2.2.3 Reti Neurali Artificiali

Le prestazioni degli algoritmi di Machine Learning più semplici dipendono in gran parte dalla rappresentazione dei dati (sottosezione 2.2.1) che ricevono [21]. Si consideri il contesto della diagnosi precoce dell'Alzheimer tramite tomografie computerizzate (TAC). Un algoritmo di Machine Learning semplice non esamina

direttamente le immagini TAC del cervello del paziente, ma riceve informazioni strutturate fornite dal medico, come la presenza di specifiche atrofie cerebrali o riduzioni del volume dell'ippocampo, caratteristiche comuni nei pazienti affetti da Alzheimer. Queste informazioni, chiamate *Feature*, vengono utilizzate dall'algoritmo per fare previsioni sulla probabilità che il paziente sviluppi la malattia. Tuttavia, se il sistema ricevesse la TAC, costituita da migliaia di pixel non etichettati, non sarebbe in grado di fare alcuna previsione. I singoli pixel non hanno una correlazione comprensibile con l'insorgenza della malattia, pertanto l'algoritmo dipende completamente dalle Feature selezionate e fornite dal medico. In maniera piuttosto analoga, anche operazioni relativamente semplici come la classificazione delle email in "spam" o "non spam" possono risultare difficili per gli algoritmi di Machine Learning se i dati non sono rappresentati correttamente, rendendo complicato ottenere previsioni accurate. Per altri compiti risulta invece estremamente complesso determinare in anticipo quali Feature debbano essere estratte e utilizzate. Questa difficoltà hanno portato alla necessità di algoritmi più avanzati, come quelli di Deep Learning, che sono in grado di apprendere automaticamente le rappresentazioni rilevanti dai dati grezzi, senza richiedere che tali rappresentazioni siano progettate manualmente.

Il Deep Learning, facendo riferimento al testo *Deep Learning* di Ian Goodfellow et al. [22], si basa su una gerarchia di concetti, con i concetti più semplici che formano quelli più complessi. Questa struttura gerarchica viene rappresentata attraverso le *Reti Neurali Artificiali (ANN)*, che consistono in più strati di neuroni artificiali. A differenza dei modelli tradizionali, le ANN possono apprendere rappresentazioni sempre più astratte dei dati, permettendo loro di risolvere compiti complessi come il riconoscimento delle immagini e la traduzione automatica. Questa evoluzione ha rappresentato un cambiamento radicale nell'Intelligenza Artificiale. I sistemi di Deep Learning non richiedono più la programmazione esplicita di regole complesse, ma acquisiscono la conoscenza necessaria attraverso l'addestramento su grandi quantità di dati, permettendo loro di generalizzare meglio anche in compiti difficili da formalizzare.

Le ANN sono alla base del Deep Learning e rappresentano un modello computazionale ispirato alla struttura e al funzionamento del cervello umano. Sono costituite da strati di unità di elaborazione chiamate neuroni artificiali, che sono organizzate in

un'architettura a più livelli: uno strato di input, uno o più strati nascosti e uno strato di output. Ciascun neurone presente in un determinato strato è collegato ai neuroni dello strato successivo tramite dei pesi, che rappresentano l'importanza delle informazioni trasmesse [21]. Per guidare il processo di addestramento del modello viene utilizzata una *Funzione di Perdita*, che quantifica la differenza tra il valore predetto dal modello e il valore reale osservato. L'obiettivo principale è minimizzare questa differenza, in quanto essa rappresenta una delle principali valutazioni delle prestazioni del modello. Nell'immagine qui riportata è mostrato un esempio di architettura di una ANN con uno strato di input composto da 8 nodi, due strati nascosti con rispettivamente 10 e 8 nodi, e uno strato di output con un singolo nodo.

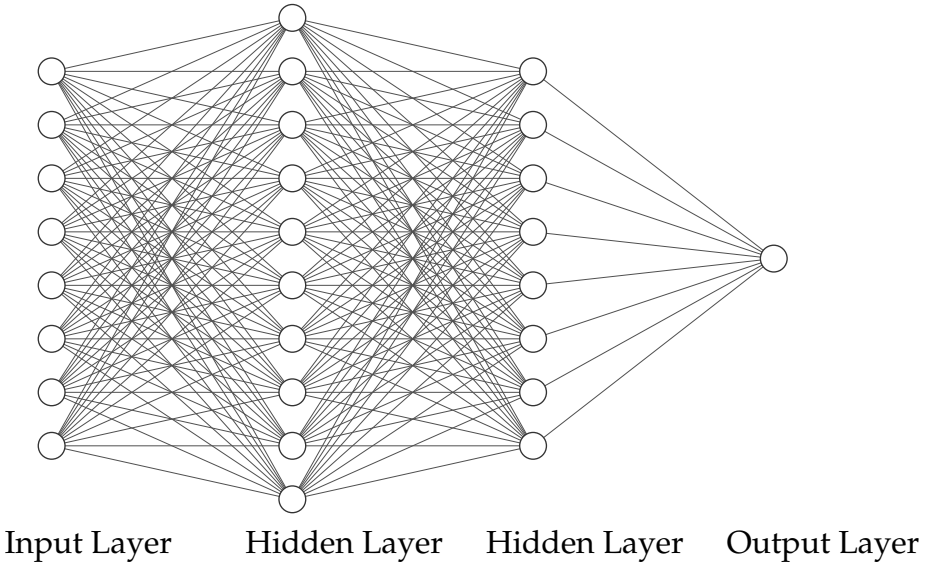

Figura 2.2: Esempio di architettura di una ANN

L'elaborazione delle informazioni in una ANN si basa sulla trasformazione dei dati attraverso questi strati, dove ogni neurone calcola una somma pesata degli input e applica una *Funzione di Attivazione* non lineare. Due delle Funzioni di Attivazione più comuni nelle ANN sono la *Sigmoide* e la *ReLu (Rectified Linear Unit)*.

Sigmoide La funzione Sigmoide è definita come:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.2.1}$$

Questa funzione mappa ogni input in un intervallo compreso tra 0 e 1, è quindi particolarmente utile per problemi di classificazione binaria. Tuttavia,

una delle sue limitazioni è il *Vanishing Gradient Problem*, in cui i gradienti nelle fasi successive dell'addestramento diventano molto piccoli, rallentando l'aggiornamento dei pesi.

ReLu La funzione ReLu è definita come:

$$f(z) = max(0, z) \tag{2.2.2}$$

Ciò significa che restituisce l'input se è positivo e zero altrimenti. La funzione ReLU è comunemente utilizzata negli ultimi anni perché i problemi di *Vanishing Gradient* sono meno frequenti rispetto alle Funzioni di Attivazione Sigmoide che si saturano in entrambe le direzioni. Uno dei problemi principali della ReLu è che può presentare il *Dying ReLU Problem*, esso si verifica quando i neuroni ricevono input negativi o nulli per un lungo periodo di tempo, dato che la funzione ReLU restituisce 0 per input negativi, i neuroni possono entrare in uno stato in cui non producono alcun output (sono "morti") e non partecipano più al processo di apprendimento, perché i gradienti non possono più fluire attraverso di loro per aggiornarne i pesi.

L'aggiornamento dei pesi della Rete avviene in due fasi principali: il *Forward Pass*, in cui i dati di input attraversano la Rete e generano una previsione, e la *Backpropagation*, che propaga l'errore dalla fine della Rete (strato di output) fino agli strati iniziali (input). Sulla base degli studi di *Yann LeCun* et al. in *Deep learning* [21], verrà fornita una spiegazione sintetica del funzionamento di tale processo, prendendo in considerazione una ANN con due strati nascosti.

Fase di Forward Pass In una ANN con due strati nascosti, ogni unità riceve degli input x_i. Questi vengono moltiplicati per i rispettivi pesi w_{ij} e sommati per ottenere il valore netto di input di ogni nodo del livello successivo:

$$z_j = \sum_{i \in Input} w_{ij} x_i \tag{2.2.3}$$

Il nodo j nello strato nascosto *H1* riceverà z_j come input netto. Dopo aver calcolato la somma pesata degli input, si applica una Funzione di Attivazione per ottenere l'output y_j:

$$y_j = f(z_j) \tag{2.2.4}$$

Questo è il valore di output per il nodo j nello strato *H1*. Successivamente si ha la propagazione verso il secondo strato nascosto, dove gli output del primo strato nascosto vengono utilizzati come input per il successivo strato *H2*, con i relativi pesi w_{jk}

$$z_k = \sum_{j \in H1} w_{jk} y_j \tag{2.2.5}$$

Il valore netto viene nuovamente attivato con una funzione non lineare per ottenere l'output y_k del secondo strato nascosto *H2*. L'output degli strati nascosti viene infine utilizzato per calcolare l'output della Rete con pesi w_{kl}:

$$z_l = \sum_{k \in H2} w_{kl} y_k \tag{2.2.6}$$

L'output finale è $y_l = f(z_l)$, dove f è la Funzione di Attivazione finale.

Calcolo dell'Errore Per il calcolo del gradiente dell'errore nello strato di output, si utilizza la derivata parziale della Funzione di Perdita rispetto all'output y_l. L'errore per l'unità l dell'output è dato dalla differenza tra l'output y_l e il valore target t_1:

$$\frac{\partial E}{\partial y_l} = y_l - t_l \tag{2.2.7}$$

Successivamente, il gradiente dell'errore rispetto all'input netto z_l all'unità l è dato dalla derivata parziale della Funzione di Attivazione f rispetto all'input netto, moltiplicata per il gradiente dell'errore rispetto all'output:

$$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l} \frac{\partial y_l}{\partial z_l} \tag{2.2.8}$$

Fase di Backpropagation Dopo aver calcolato l'errore nello strato di output, l'errore viene propagato indietro attraverso gli strati nascosti. Per ciascun nodo k in uno strato nascosto, l'errore è calcolato sommando l'errore proveniente dagli strati successivi. In particolare, l'errore per il nodo k dello strato nascosto *H2* è dato da:

$$\frac{\partial E}{\partial z_k} = \sum_{l \in out} w_{kl} \frac{\partial E}{\partial z_l} \tag{2.2.9}$$

Lo stesso processo di Backpropagation viene applicato per calcolare l'errore nei nodi j dello strato nascosto *H1*, sommando l'errore dai nodi k di *H2*

$$\frac{\partial E}{\partial z_j} = \sum_{k \in H2} w_{jk} \frac{\partial E}{\partial z_k} \tag{2.2.10}$$

Aggiornamento dei pesi Una volta calcolati i gradienti per tutte le unità nella Rete, si può aggiornare ogni peso w_{ij} utilizzando una regola di ottimizzazione come il *Gradient Descent*. Questo processo viene gestito da un *Optimizer*, che determina in che modo i pesi devono essere modificati per ridurre l'errore complessivo del modello. Consideriamo l'utilizzo del Gradient Descent:

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}} \tag{2.2.11}$$

Dove:

w_{ij} = Peso che collega l'unità i all'unità j

η = Tasso di Apprendimento (Learning Rate)

$\frac{\partial E}{\partial w_{ij}}$ = Gradiente dell'errore rispetto al peso w_{ij}

Grazie a questo meccanismo iterativo, una ANN è in grado di apprendere e migliorare le sue prestazioni, adattandosi progressivamente ai dati forniti e rendendo possibile l'addestramento di modelli complessi in grado di risolvere una vasta gamma di problemi.

Nella formula 2.2.11 è stato menzionato il *Tasso di Apprendimento (Learning Rate - LR)*, il Learning Rate viene utilizzato dall'Optimizer e controlla la dimensione dei passi che l'algoritmo compie verso il minimo della Funzione di Perdita. Determina quindi quanto velocemente o lentamente il modello aggiorna i suoi pesi in risposta all'errore osservato durante l'addestramento.

- Se il Learning Rate è troppo alto, l'Optimizer potrebbe "saltare" oltre il minimo, portando a una convergenza instabile o addirittura a una divergenza, dove l'errore di addestramento inizia ad aumentare invece di diminuire.

- Al contrario, se il Learning Rate è troppo basso, l'addestramento diventa molto lento perché l'Optimizer fa passi molto piccoli. Ciò può portare a un lungo tempo di addestramento e c'è il rischio che l'addestramento si arresti in un minimo locale anziché globale, a seconda della natura della Funzione di Perdita.

È quindi importante mantenere il Learning Rate sufficientemente basso da evitare oscillazioni eccessive ma abbastanza alto da garantire una convergenza efficiente senza rimanere bloccati in minimi locali o richiedere troppe iterazioni.

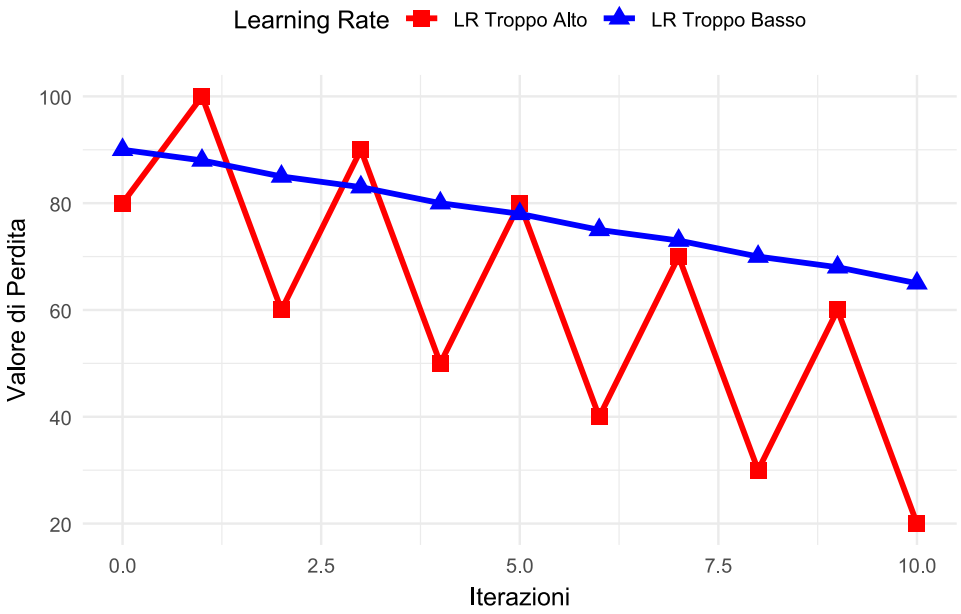

Figura 2.3: Effetto del Learning Rate sull'ottimizzazione

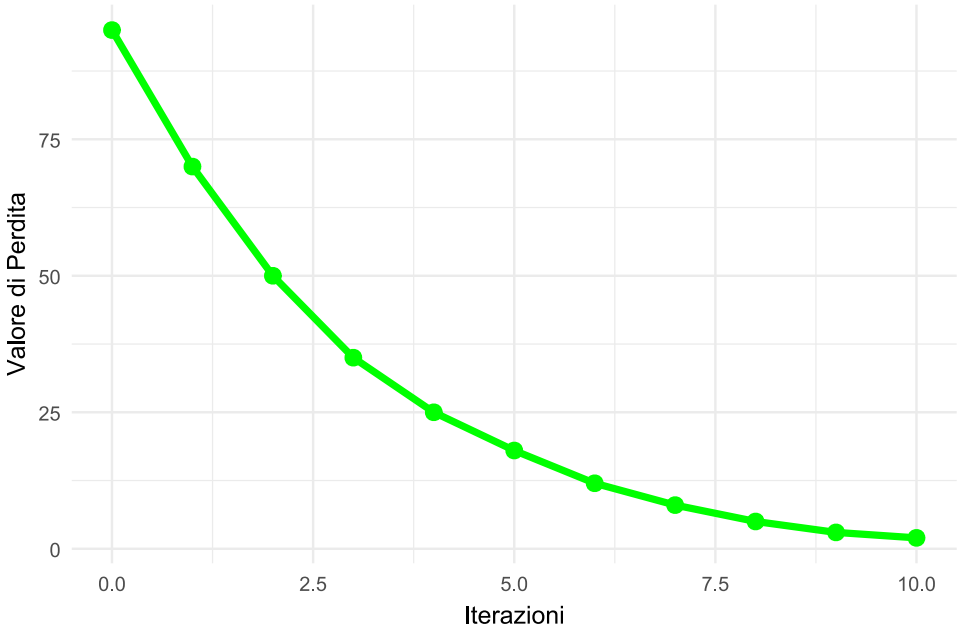

Figura 2.4: Convergenza con Learning Rate ideale

2.3 Natural Language Processing

Negli ultimi anni, il *Natural Language Processing (NLP)* ha ottenuto un ruolo centrale nell'IA, permettendo lo sviluppo di modelli in grado di comprendere e generare il linguaggio umano. Il NLP si occupa di trasformare il linguaggio naturale in una forma che possa essere processata dalle macchine, permettendo applicazioni come la traduzione automatica, l'analisi del sentiment e la generazione di testo. Grazie agli avanzamenti nei modelli di linguaggio su larga scala, oggi il NLP ha raggiunto un livello di avanzamento tale che i testi generati da questi modelli risultano spesso indistinguibili da quelli scritti da esseri umani [5].

Le difficoltà principali nel NLP derivano dalla complessità del linguaggio umano, che presenta ambiguità, variazioni contestuali e strutture sintattiche articolate [23]. A differenza degli approcci tradizionali basati su regole, i modelli moderni di NLP utilizzano rappresentazioni vettoriali (embedding) per mappare le parole in uno spazio continuo, catturando le relazioni semantiche. Grazie a queste rappresentazioni vettoriali, i modelli di NLP possono gestire la complessità del linguaggio umano in modo più efficace, superando i limiti degli approcci basati su regole rigide.

2.3.1 Principali Applicazioni del NLP

Il NLP ha portato a una vasta gamma di applicazioni che spaziano dai sistemi di comprensione del linguaggio a modelli in grado di soddisfare diversi tipi di richieste. Tali progressi hanno reso possibile l'uso del NLP in molteplici settori, che vanno dal marketing all'assistenza sanitaria, fino ai servizi finanziari.

Traduzione Automatica La traduzione automatica è una delle applicazioni più conosciute del NLP. Inizialmente sviluppata utilizzando modelli basati su regole e successivamente modelli statistici, questa tecnologia ha compiuto un enorme salto qualitativo con l'introduzione delle Reti Neurali Ricorrenti (RNN) utilizzate da Google Translate [24]. Successivamente, l'introduzione dei Transformer [1] ha permesso un ulteriore miglioramento della qualità delle traduzioni, rendendole più fluide e contestuali e ponendo le basi per l'evoluzione verso i moderni modelli di linguaggio di grandi dimensioni [5].

Analisi del Sentiment L'analisi del Sentiment è una tecnica utilizzata per estrarre opinioni e stati d'animo da testi, come recensioni dei clienti o commenti sui social media. Questa applicazione trova particolare uso nelle strategie di marketing, in quanto permette alle aziende di comprendere le percezioni dei consumatori sui loro prodotti [25].

Chatbot e Assistenti Virtuali I chatbot e gli assistenti virtuali, come Siri di Apple, Alexa di Amazon, e Google Assistant, sono diventati strumenti sempre più popolari per l'interazione tra macchine e utenti [26]. Grazie ai progressi dei modelli di NLP, i chatbot sono ora in grado di comprendere e rispondere a una gamma di richieste complesse, migliorando l'esperienza dell'utente e aumentando l'efficienza delle interazioni [27].

Riconoscimento di Entità Nominate (NER) Il Riconoscimento di Entità Nominate (NER) è una tecnica che permette di identificare e classificare entità, come nomi di persone, luoghi, e organizzazioni, all'interno di testi. Questo metodo è cruciale per applicazioni come l'estrazione di informazioni e l'analisi di documenti aziendali [28]. Modelli avanzati come SpaCy e BERT sono spesso utilizzati per migliorare la precisione del NER, permettendo di estrarre informazioni strutturate da testi non strutturati [29].

Riassunto Automatico del Testo Il riassunto automatico è diventato uno strumento importante per estrarre i contenuti rilevanti da documenti lunghi, come articoli scientifici e libri. Modelli come T5 [6] e BART [30] hanno raggiunto risultati all'avanguardia in questa applicazione, facilitando l'accesso rapido alle informazioni.

Question Answering (QA) Negli ultimi anni, il campo del Question Answering (QA) ha compiuto enormi passi avanti grazie all'evoluzione dei modelli di linguaggio di grandi dimensioni, in particolare con la famiglia di modelli GPT, che ha raggiunto livelli di comprensione del linguaggio tali da permettere risposte a domande in modo quasi indistinguibile da un esperto umano [5]. Uno degli aspetti chiave per sfruttare queste capacità è il Prompt Engineering, una tecnica che consente di guidare il modello verso risposte più accurate e

inerenti formulando strategicamente l'input [31]. Grazie al Prompt Engineering, i modelli possono essere adattati a una varietà di task di QA, permettendo di ottenere risposte dettagliate e rilevanti senza la necessità di ulteriori fasi di addestramento.

Moderazione dei Contenuti Il NLP è diventato fondamentale per la moderazione dei contenuti sui social media, in particolare per il rilevamento delle fake news. Con l'aumento della disinformazione, i modelli di NLP sono stati impiegati per identificare testi ingannevoli o offensivi. Studi recenti hanno utilizzato BERT e RoBERTa per rilevare pattern linguistici indicativi di disinformazione o contenuti dannosi [32].

2.3.2 Rappresentazione del testo

Uno dei primi passi fondamentali per l'elaborazione del linguaggio naturale è rappresentare il testo in una forma numerica che i modelli di IA possano comprendere. Esistono diverse tecniche per rappresentare il testo, ognuna con vantaggi e limitazioni, e queste tecniche si sono evolute nel tempo per catturare meglio il significato e il contesto linguistico.

Tra le prime tecniche utilizzate per rappresentare il testo vi è il *Bag of Words (BoW)*, in cui ogni documento viene convertito in un vettore basato sulla frequenza delle parole. Sebbene semplice, BoW ignora l'ordine delle parole e le relazioni semantiche tra di esse, limitandone l'efficacia in compiti avanzati di NLP. Per migliorare BoW, è stato introdotto *TF-IDF (Term Frequency-Inverse Document Frequency)*, che riduce il peso delle parole comuni e aumenta l'importanza di quelle più distintive per ogni documento [23].

Con l'avvento delle Reti Neurali, sono state sviluppate tecniche più sofisticate come *Word2Vec* di *Mikolov* et al. (2013), che utilizza Reti Neurali per apprendere rappresentazioni vettoriali continue delle parole, dove parole con significati simili si trovano vicine nello spazio vettoriale. Word2Vec ha introdotto due modelli principali: il *Continuous Bag of Words (CBOW)*, che utilizza il contesto di parole circostanti per prevedere una parola centrale, e *Skip-Gram*, che usa una parola per prevedere il contesto circostante. Questi modelli hanno dimostrato di catturare relazioni semantiche

come analogie, permettendo rappresentazioni più ricche del testo [33]. Parallelamente, è stato sviluppato *GloVe (Global Vectors for Word Representation)* da *Pennington* et al. (2014), un metodo che sfrutta le co-occorrenze globali delle parole in un corpus per creare vettori di parole che catturano relazioni semantiche [34]. GloVe ha ottenuto risultati notevoli in compiti come la classificazione testuale e il riconoscimento di entità nominate.

Più recentemente, i modelli basati su *Transformer* hanno introdotto embeddings contestuali che superano i limiti delle rappresentazioni statiche come Word2Vec e GloVe, offrendo rappresentazioni più dinamiche e sensibili al contesto delle parole. Questa innovazione ha permesso di migliorare in modo significativo l'efficacia dei modelli in una vasta gamma di compiti linguistici, dimostrando la capacità di adattarsi a contesti complessi attraverso il pre-addestramento su grandi quantità di dati e successivo *Fine-Tuning* (sottosezione 2.4.2).

2.3.3 Encoder, Decoder ed Encoder-Decoder

Nell'ambito del Deep Learning e del NLP, le architetture *Encoder*, *Decoder* ed *Encoder-Decoder* sono fondamentali per elaborare e generare testo, adattandosi a compiti diversi in base alle specifiche esigenze di comprensione e generazione del linguaggio.

Encoder Un Encoder è una Rete Neurale che prende in input una sequenza di token e la trasforma in una rappresentazione latente (embedding), che contiene le informazioni essenziali del testo. Questo processo permette di catturare il contesto e le relazioni semantiche tra i token, offrendo una rappresentazione arricchita utile per compiti di comprensione, come la classificazione e il riconoscimento di entità nominate. Modelli come BERT (Bidirectional Encoder Representations from Transformers) e RoBERTa (A Robustly Optimized BERT Pretraining Approach) sono esempi di architetture Encoder progettate per compiti di comprensione del linguaggio [35, 36].

Decoder Un Decoder è una Rete Neurale utilizzata per generare testo a partire da una rappresentazione latente. Generalmente, il Decoder opera in modalità autoregressiva, generando un token alla volta e utilizzando i token precedenti

per prevedere il successivo. Questo approccio è efficace per task di generazione di testo, come il completamento delle frasi e la risposta a prompt. Un esempio di modello basato su decoder è GPT (Generative Pre-trained Transformer), che è particolarmente adatto alla generazione di testo in modo coerente e contestuale [37, 5].

Encoder-Decoder L'architettura Encoder-Decoder combina entrambe le strutture, rendendola ideale per compiti che richiedono sia la comprensione che la generazione del linguaggio, come la traduzione automatica e il riassunto di testi. In questa architettura, l'Encoder trasforma l'input in una rappresentazione latente, che viene poi passata al Decoder per generare l'output. Modelli come T5 (Text-to-Text Transfer Transformer) e BART (Bidirectional and Auto-Regressive Transformers) utilizzano questa struttura per eccellere in task di trasformazione del testo [6, 30].

Queste architetture, combinate con i meccanismi di *Attention*, hanno reso possibili modelli di NLP specializzati e potenti, migliorando la capacità di comprendere il contesto linguistico e produrre risposte coerenti e precise [5].

2.3.4 Sfide del NLP

Nonostante i notevoli progressi nel campo del NLP, in gran parte dovuti ai modelli di Deep Learning e agli attuali LLM, persistono ancora sfide complesse. Queste problematiche derivano principalmente dalla natura complessa e sfaccettata del linguaggio umano [38].

Una delle sfide più rilevanti è la comprensione contestuale a lungo termine. Sebbene modelli avanzati come BERT, GPT e T5 abbiano migliorato significativamente la capacità di catturare le relazioni contestuali tra parole e frasi, incontrano ancora difficoltà quando si tratta di comprendere contesti molto estesi o strutture narrative articolate. In particolare, la limitazione della lunghezza della sequenza nei modelli Transformer influisce sia sull'input che sull'output, riducendo la capacità di mantenere dipendenze a lungo termine. La struttura di Attention utilizzata nei Transformer comporta una complessità che cresce quadraticamente con l'aumento della lunghezza della sequenza, limitando il modello a gestire efficacemente solo sequenze entro un

certo limite, come i 512 token nei modelli originali descritti da *Vaswani* et al. [1]. A fronte di queste limitazioni, gli studi di *Tay* et al. [39] esplorano diverse varianti dei Transformer progettate per ridurre la complessità computazionale della struttura di Attention, introducendo meccanismi alternativi che consentono di gestire sequenze più lunghe in modo efficiente, senza perdere rilevanza nel contesto. Tuttavia, queste soluzioni comportano compromessi, come una ridotta precisione nelle relazioni tra i token e la necessità di approssimazioni, che possono compromettere la qualità della rappresentazione del contesto globale.

Un'altra sfida cruciale è rappresentata dai bias e pregiudizi nei modelli di NLP. Essendo addestrati su ampi dataset estratti da Internet o da altre fonti umane, questi modelli tendono a riprodurre i bias culturali e sociali presenti nei dati di addestramento. Tali pregiudizi possono portare a risultati non etici o discriminatori [40]. La mitigazione di questi bias è attualmente una priorità, richiedendo lo sviluppo di metodologie per rilevare e ridurre pregiudizi all'interno dei dati e delle rappresentazioni generate dai modelli [41].

Anche le limitazioni nella generalizzazione rappresentano una questione rilevante. Sebbene i modelli di NLP dimostrino alte prestazioni nei loro domini di addestramento, talvolta non riescono a trasferire questa capacità a contesti nuovi o dati sconosciuti. Questo problema è evidente quando i modelli producono frasi che, pur essendo formalmente corrette, risultano semanticamente inadeguate. La capacità di adattarsi a dati eterogenei e variegati è quindi cruciale per migliorare l'efficacia dei modelli, e approcci come il *Multi-Task Learning* e il *Domain Adaption* mirano a migliorare la generalizzazione apprendendo informazioni comuni tra più task o adattando i modelli a specifici contesti di utilizzo [9].

Infine, esistono difficoltà legate all'interpretabilità, o *Explainability*, dei modelli. Spesso percepiti come "scatole nere", questi sistemi rendono complessa la comprensione delle ragioni alla base delle loro risposte. Questa mancanza di trasparenza rappresenta un ostacolo significativo per l'adozione in settori regolamentati come la medicina e la finanza, dove è cruciale giustificare e spiegare le decisioni prese [42]. Nonostante i progressi compiuti, rendere interpretabili i modelli di Deep Learning complessi resta una sfida aperta [43].

In conclusione, le sfide affrontate nel campo del NLP evidenziano l'importanza di

non concentrarsi esclusivamente sui progressi tecnologici, ma anche di promuovere un approccio etico e trasparente. È fondamentale sviluppare sistemi di NLP che siano non solo efficaci, ma anche in grado di operare in conformità con i valori sociali, garantendo così un utilizzo responsabile e affidabile della tecnologia.

2.4 Large Language Models

Con l'avanzamento delle Reti Neurali Artificiali e l'evoluzione del Deep Learning, sono stati sviluppati modelli su larga scala noti come *Large Language Models (LLM)*. Gli LLM rappresentano un ulteriore passo avanti nell'ambito dell'IA, in quanto sfruttano una quantità massiva di parametri e dati di addestramento per migliorare la comprensione e la generazione del testo. L'enorme quantità di parametri è uno degli aspetti distintivi degli LLM, mentre i modelli di Deep Learning tradizionali possono avere milioni di parametri, gli LLM più avanzati operano con miliardi o addirittura bilioni di parametri [44]. I parametri rappresentano le componenti del modello che vengono apprese dai dati durante la fase di addestramento; in altre parole, sono le parti del modello che vengono modificate per ridurre l'errore di previsione. Maggiore è il numero di parametri, più è probabile che il modello riesca a cogliere contesti complessi e sfumature nei dati [45]. Questi modelli avanzati, esemplificati da GPT-3, BERT e i loro derivati, hanno ottenuto notevole attenzione e riconoscimento per la loro straordinaria capacità di comprendere e produrre linguaggio umano. Negli ultimi anni, sono stati sviluppati modelli ancora più potenti, come GPT-4o, l'ultima versione di GPT, e il modello Switch Transformer da 1,6 bilioni di parametri [46, 44]. L'impatto degli LLM sta rendendo sempre più evidente come ogni ambito della società — dall'economia alla sanità, dall'educazione alla scienza — sarà trasformato in modo radicale, aprendo la strada a un'era di automazione intelligente e di innovazione senza precedenti, in cui l'IA non solo supporta, ma ridefinisce processi, conoscenze e modalità di interazione. D'altronde anche i modelli più sofisticati devono affrontare sfide come l'overfitting, la gestione di parole rare o mai viste e la cattura della complessa struttura del linguaggio. Miglioramenti continui all'architettura e alle metodologie di addestramento sono essenziali per superare questi ostacoli e migliorare le prestazioni degli LLM [5].

2.4.1 L'architettura Transformer

I meccanismi di *Attention* sono diventati una componente fondamentale nella modellazione delle sequenze e nei modelli di trasduzione per vari compiti, permettendo di catturare le dipendenze indipendentemente dalla loro distanza nelle sequenze di input o output. Prima dell'introduzione del Transformer, i modelli basati su Reti Ricorrenti (RNN), inclusi LSTM e GRU, erano ampiamente utilizzati per la loro capacità di gestire le dipendenze temporali. [47, 48]. Tuttavia, questi modelli soffrivano di limitazioni, come la difficoltà nel parallelizzare il processo di addestramento e nel gestire dipendenze a lungo termine in modo efficace [49]. Sulla base di queste sfide è stata introdotta l'architettura Transformer che sfrutta interamente il meccanismo di *Self-Attention* per modellare le dipendenze tra elementi in una sequenza, superando così le limitazioni delle Reti Ricorrenti.

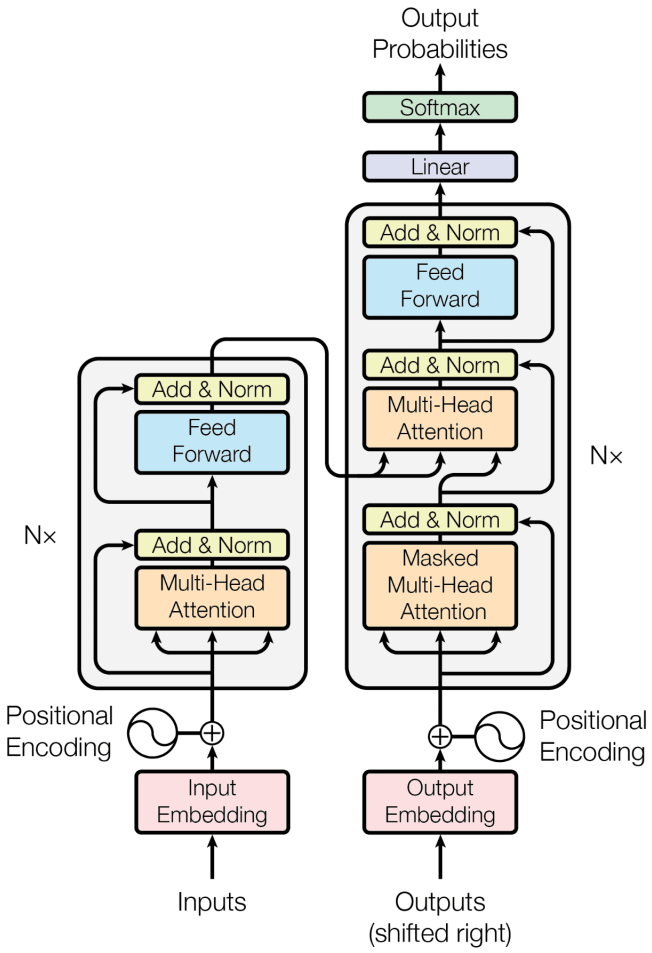

Figura 2.5: Architettura Transformer ([1])

Questa architettura si è affermata come la base dei moderni LLM grazie alla sua capacità di processare dati in parallelo e di gestire grandi quantità di testo, rendendola ideale per applicazioni avanzate di *Natural Language Processing (NLP)* [50]. Nella figura 2.5 è mostrata l'architettura Transformer, alcuni dei componenti principali di questa architettura, sono brevemente descritti di seguito, sulla base delle spiegazioni fornite da *Vaswani* et al. [1]. L'architettura Transformer utilizza una struttura Encoder-Decoder (sottosezione 2.3.3) utilizzando uno stack di Self-Attention e layer completamente connessi per entrambi i componenti, Encoder e Decoder. Questo design elimina la necessità delle Reti Ricorrenti, consentendo al modello di elaborare le sequenze in parallelo e sfruttare il contesto globale tramite il meccanismo di Attention. L'Encoder è costituito da uno stack di 6 layer identici, dove ogni layer è composto da due sub-layer: un meccanismo di Multi-Head Self-Attention e una rete Feed-Forward completamente connessa, applicata separatamente a ciascuna posizione della sequenza. Ogni sub-layer nell'Encoder è circondato da una connessione residuale e seguito da una normalizzazione layer-wise [51]. Tutti i sub-layer, inclusi quelli di embedding, producono output di dimensione fissa 512. Il Decoder è anch'esso costituito da uno stack di 6 layer identici. Oltre ai due sub-layer presenti in ciascun layer dell'Encoder, il Decoder inserisce un terzo sub-layer, che esegue un'operazione di Multi-Head Attention sull'output dello stack dell'Encoder. Come nell'Encoder, ogni sub-layer è circondato da una connessione residuale e seguito da una normalizzazione layer-wise. Il masking applicato al Self-Attention del decoder garantisce che la generazione del token in una determinata posizione dipenda solo dai token generati fino a quella posizione, rendendo il processo autoregressivo.

L'Encoder e il Decoder del Transformer si basano sul meccanismo di Attention per catturare il contesto e le relazioni tra i token in una sequenza. Per comprendere meglio come questi processi contribuiscono alla capacità del Transformer di rappresentare e generare testo, è utile approfondire il funzionamento della Scaled Dot-Product Attention e della Multi-Head Attention, che sono alla base dell'architettura del Transformer. Il concetto di Attention può essere descritto come una funzione che associa una query e una serie di coppie chiave-valore a un output. In questa struttura, query, chiavi, valori e output sono tutti vettori. Il risultato dell'operazione di Attention è una somma pesata dei valori, dove il peso assegnato a ciascun valore viene calcolato

in base a una funzione di compatibilità tra la query e la corrispondente chiave.

Scaled Dot-Product Attention La Scaled Dot-Product Attention riceve in input le query e le chiavi di dimensione d_k, e i valori di dimensione d_v. La procedura consiste nel calcolare i prodotti scalari tra le query e tutte le chiavi, dividere ciascun prodotto per $\sqrt{d_k}$, e quindi applicare una funzione *Softmax* per ottenere i pesi dei valori.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4.1}$$

Questa forma di Attention risulta molto efficiente, soprattutto perché può essere implementata come una moltiplicazione di matrici ottimizzata. L'Additive Attention, invece, calcola la funzione di compatibilità utilizzando una rete Feed-Forward con un singolo strato nascosto. Sebbene entrambe le forme abbiano complessità teorica simile, la Scaled Dot-Product Attention è preferita rispetto a quella additiva in quanto risulta più veloce e meno onerosa in termini di spazio, specialmente per dimensioni più grandi di d_k.

Multi-Head Attention Il meccanismo di Multi-Head Attention consente al modello di eseguire più operazioni di Attention in parallelo, ognuna con una diversa proiezione lineare delle query, chiavi e valori. In pratica, le query, chiavi e valori vengono proiettati h volte su diversi sotto-spazi tramite trasformazioni lineari, e su ciascuna di queste proiezioni viene eseguita l'Attention in parallelo. I risultati delle diverse operazioni di Attention vengono poi concatenati e nuovamente proiettati per ottenere l'output finale.

La formula per la Multi-Head Attention è:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^O \tag{2.4.2}$$

Dove:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4.3}$$

Con $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ e $W^O \in \mathbb{R}^{hd_v \times d_{model}}$

Il vantaggio della Multi-Head Attention è che consente al modello di attendere su diverse parti della sequenza in parallelo, migliorando la capacità di rappresentare relazioni complesse tra i token all'interno della sequenza.

Esistono tre applicazioni principali dei meccanismi di Multi-Head Attention nel Transformer:

1. Self-Attention nell'Encoder-Decoder: Nei layer di Attention Encoder-Decoder, le query provengono dal layer precedente del Decoder, mentre le chiavi e i valori derivano dall'output dell'Encoder. Questo tipo di Attention permette a ogni posizione nel Decoder di concentrarsi su tutte le posizioni nella sequenza di input, facilitando la trasmissione di informazioni dal contesto di input alla generazione dell'output.

2. Self-Attention nell'Encoder: I layer di Self-Attention nell'Encoder consentono a ogni posizione di "attendere" su tutte le altre posizioni della sequenza di input, migliorando la capacità del modello di catturare dipendenze e relazioni all'interno del contesto di input. In questo caso, le query, le chiavi e i valori provengono tutti dallo stesso layer dell'Encoder, e ogni posizione nell'Encoder può accedere a tutte le posizioni del layer precedente.

3. Self-Attention nel Decoder: I layer di Self-Attention nel Decoder funzionano in modo simile, ma con una differenza importante: ogni posizione nel Decoder può "attendere" solo sulle posizioni fino a quella corrente, garantendo che il modello non utilizzi informazioni future durante la generazione autoregressiva. Questo viene realizzato applicando un masking (impostando a $-\infty$) ai valori di Attention per bloccare connessioni non valide e preservare la sequenzialità autoregressiva.

Grazie a questa struttura, oggi è possibile affrontare compiti linguistici complessi come la traduzione automatica, il riassunto e il question answering in modo altamente efficiente. La flessibilità del Transformer e la sua capacità di adattarsi a diversi contesti linguistici pongono le basi per un utilizzo del linguaggio sempre più naturale da parte delle macchine, aprendo nuove frontiere nell'ambito del NLP [52].

2.4.2 Fine-Tuning e Transfer Learning

Una delle caratteristiche più potenti degli LLM è la loro capacità di adattarsi a vari compiti attraverso il *Transfer Learning* e il *Fine-Tuning* [53]. Questi modelli vengono inizialmente pre-addestrati su vasti dataset generici, acquisendo una comprensione ampia e versatile dei dati.

Il Transfer Learning è una tecnica di apprendimento automatico che prevede il riutilizzo di un modello pre-addestrato su un nuovo compito simile, ma non identico, a quello su cui il modello è stato inizialmente addestrato. L'idea principale è che le conoscenze acquisite durante l'addestramento su un compito possano essere trasferite e utilizzate per migliorare le prestazioni su un altro compito. Il modello viene addestrato su un grande dataset, spesso con compiti generali, e successivamente viene adattato (Fine-Tuned) su un dataset specifico del nuovo compito, che può essere più piccolo e mirato rispetto al dataset utilizzato per il pre-addestramento.

In linea con questo approccio, *Howard* e *Ruder* (2018) hanno proposto il metodo *Universal Language Model Fine-tuning (ULMFiT)* [54], che ha dimostrato l'efficacia del Transfer Learning nel NLP. Nel loro studio, ULMFiT ha superato lo stato dell'arte in sei compiti di classificazione testuale, riducendo l'errore di un 18%-24% nella maggior parte dei dataset. Inoltre, con solo 100 esempi etichettati, ULMFiT ha raggiunto prestazioni comparabili a quelle ottenute con 100 volte più dati utilizzando metodi tradizionali. Questi risultati evidenziano come il Fine-Tuning e il Transfer Learning possano migliorare significativamente l'efficienza e l'efficacia dei modelli di linguaggio, permettendo loro di eccellere in compiti specifici con una quantità limitata di dati etichettati.

L'efficacia del Transfer Learning e del Fine-Tuning è stata ulteriormente dimostrata da modelli come BERT di *Devlin* et al. (2018), che ha utilizzato un pre-addestramento Non Supervisionato seguito da un Fine-Tuning su task specifici, ottenendo risultati di stato dell'arte in molteplici compiti di NLP, come il Question Answering e il riconoscimento di entità nominate [35]. Allo stesso modo, GPT di *Radford* et al. (2018) ha evidenziato come un modello di linguaggio addestrato su enormi dataset possa essere Fine-Tuned con successo per adattarsi a diversi task senza richiedere enormi quantità di dati etichettati, dimostrando la scalabilità del

Transfer Learning [37]. Più recentemente, T5 di *Raffel* et al. (2020) ha esplorato un approccio unificato di Text-to-Text, mostrando che un modello pre-addestrato su task generali può essere facilmente adattato a vari task di NLP attraverso il Fine-Tuning [6]. Questi studi supportano l'idea che il Transfer Learning e il Fine-Tuning non solo migliorano le prestazioni dei modelli, ma anche la loro capacità di adattamento a nuovi contesti con quantità limitate di dati.

2.4.3 Sfide degli LLM

Nel contesto dell'adozione e dello sviluppo degli LLM, emergono diverse sfide che non riguardano unicamente la complessità computazionale, ma anche la gestione di risorse limitate e la sostenibilità a lungo termine. Tra le problematiche principali vi sono anche la gestione dei bias nei dati e l'impatto ambientale, tutte questioni che influiscono significativamente sull'adozione degli LLM in scenari reali e regolamentati.

La questione dell'uso ottimale delle risorse emerge in modo significativo durante il Fine-Tuning. Gli LLM richiedono ingenti risorse per il loro addestramento e adattamento a compiti specifici. Tecniche come la *Low-Rank Adaptation (LoRA)* [55], introdotte per ridurre la complessità computazionale, rappresentano una soluzione importante per affrontare questi limiti. LoRA consente di ridurre il numero di parametri da ottimizzare durante il Fine-Tuning, concentrandosi solo su adattamenti a bassa dimensione delle matrici di peso, senza dover riaddestrare l'intero modello. Questo approccio migliora l'efficienza e la flessibilità del Fine-Tuning, permettendo di adattare modelli pre-addestrati a risorse limitate senza compromettere eccessivamente le prestazioni. LoRA è particolarmente utile anche per la gestione della memoria. Poiché solo una parte limitata del modello viene effettivamente modificata, i requisiti di memoria per il Fine-Tuning si riducono significativamente, consentendo di adattare modelli pre-addestrati anche su dispositivi con risorse limitate. Come dimostrato da *Hu* et al. (2021), l'utilizzo di LoRA ha permesso di ridurre il consumo di memoria GPU fino al 66% (e.g., nel caso del modello GPT-3 da 175 miliardi di parametri, la VRAM è stata ridotta da 1.2 TB a 350 GB). Inoltre, LoRA ha consentito una drastica riduzione dello spazio di archiviazione necessario per i checkpoint, pas-

sando da 350 GB a soli 35 MB. In ambienti in cui la memoria e la potenza di calcolo sono risorse scarse, LoRA rappresenta una soluzione particolarmente vantaggiosa, poiché consente di adattare modelli di grandi dimensioni in modo più scalabile e sostenibile.

Un'altra tecnica cruciale per ottimizzare gli LLM è la *Quantizzazione* [56]. La Quantizzazione riduce la precisione dei numeri utilizzati nei parametri del modello, passando per esempio da 32 bit a 8 bit. Questo processo riduce la memoria necessaria per i modelli e ne aumenta l'efficienza, dimostrandosi particolarmente utile in contesti in cui le risorse di calcolo e memoria sono limitate. Il funzionamento della Quantizzazione si basa sulla trasformazione dei pesi e delle attivazioni da valori a virgola mobile a valori interi, mediante una scala che preserva il range di valori originali. Per minimizzare l'errore di Quantizzazione, si utilizza un "zero-point" che allinea lo zero reale con il valore più vicino nella rappresentazione intera, migliorando la gestione delle attivazioni che possono avere valore zero. In questo modo, il modello può eseguire l'inferenza utilizzando solo operazioni con numeri interi. *Jacob* et al. (2017) dimostrano che questa tecnica permette di ottenere un'inferenza efficiente con un minor consumo di risorse, senza sacrificare in modo rilevante la precisione del modello.

Per quanto riguarda l'impatto ambientale, uno dei problemi più rilevanti è l'enorme consumo energetico associato all'addestramento degli LLM. Secondo uno studio del 2022 condotto da *Luccioni* et al. [57], l'addestramento di grandi modelli come GPT-3, composto da 175 miliardi di parametri, ha comportato un consumo energetico stimato di circa 1.287 MWh, con un'impronta di carbonio di 502 tonnellate di CO_2 equivalente, variabile in base alle fonti di energia utilizzate. Per contrastare ciò, sono in corso ricerche che mirano a ridurre il consumo energetico, come quelle discusse precedentemente.

Infine, come discusso nella sezione sulle sfide del NLP (sottosezione 2.3.4), la generalizzazione dei modelli, i problemi di bias e la mancanza di interpretabilità rimangono sfide significative anche per gli LLM in generale. Nonostante le elevate prestazioni in contesti di addestramento, questi modelli spesso faticano a trasferire le loro capacità a compiti e dati non presenti nel set di addestramento. Inoltre, a causa della natura dei vasti dataset utilizzati, gli LLM sono esposti ai problemi di bias

descritti, che possono introdurre pregiudizi non intenzionali nei risultati generati. La limitata interpretabilità di questi modelli rende inoltre complessa l'adozione in contesti in cui la trasparenza è essenziale. Affrontare queste sfide è cruciale per garantire l'uso responsabile e affidabile degli LLM in applicazioni pratiche.

In sintesi, sebbene le tecniche di ottimizzazione come LoRA e la Quantizzazione abbiano portato miglioramenti significativi nell'efficienza e nella gestione delle risorse per gli LLM, restano sfide cruciali che devono essere affrontate. La sostenibilità energetica, l'adattabilità a contesti diversi e l'eliminazione dei bias nei dati sono obiettivi chiave per garantire un'adozione sicura e responsabile di questi modelli su larga scala.

2.5 GenAI e Peer Review

Con l'avvento dell'*Intelligenza Artificiale Generativa (GenAI)*, si sono aperte nuove possibilità nel campo dell'automazione e della creazione di contenuti, tra cui testi e revisioni. Tuttavia, l'uso di GenAI nella Peer Review solleva questioni etiche e pratiche, specialmente in ambito accademico, dove è fondamentale garantire la qualità e l'integrità delle valutazioni. Di conseguenza, molte riviste scientifiche, come quelle di *Elsevier* [58], hanno adottato policy specifiche per regolamentarne l'uso, limitando o proibendo l'impiego di GenAI da parte di autori, revisori ed editori. Questa sezione è basata principalmente sulle linee guida di *Elsevier*, pur tenendo presente che ogni rivista potrebbe avere proprie policy specifiche che è opportuno considerare.

Per quanto riguarda gli autori, *Elsevier* consente l'uso di tecnologie di GenAI solo per migliorare la leggibilità e il linguaggio del manoscritto, sempre sotto supervisione umana. Gli autori devono rivedere attentamente i risultati e sono responsabili del contenuto. L'uso dell'IA deve essere dichiarato nel manoscritto per garantire trasparenza, e l'IA non può essere elencata come autore o co-autore. *Elsevier* vieta inoltre l'uso di GenAI per creare o modificare immagini nei manoscritti, a meno che non faccia parte del design o del metodo di ricerca, in quel caso l'uso dell'IA deve essere documentato nella sezione dei metodi. Per grafici astratti o copertine, è necessaria un'autorizzazione specifica e il rispetto dei diritti d'uso.

Per i revisori, *Elsevier* richiede che i manoscritti siano trattati come documenti confidenziali e non siano caricati su strumenti di GenAI, poiché ciò potrebbe violare la riservatezza e i diritti di proprietà degli autori. Anche i rapporti di revisione devono rimanere confidenziali e non devono essere caricati su strumenti di IA, neppure per migliorare la leggibilità.

Infine, per gli editori, *Elsevier* stabilisce nuovamente che i manoscritti sottomessi siano trattati come documenti confidenziali, vietando l'uso di GenAI per caricarli o modificarli, per evitare violazioni della privacy e dei diritti di proprietà degli autori. Questo obbligo di riservatezza si estende anche alle comunicazioni editoriali riguardanti il manoscritto.

In conclusione, l'uso di GenAI nel processo di Peer Review offre potenzialità, ma comporta anche sfide etiche e pratiche significative, soprattutto per quanto riguarda la riservatezza, la responsabilità e l'integrità delle valutazioni. Attualmente la maggior parte delle riviste, inclusa *Taylor & Francis* [59], vieta l'uso di GenAI per i revisori. Mentre GenAI può supportare aspetti secondari del lavoro accademico, come il miglioramento della leggibilità dei testi, le responsabilità critiche del processo di revisione e valutazione restano un compito degli esseri umani, a cui è richiesta una capacità di giudizio e un pensiero critico che la tecnologia attualmente non è in grado di replicare. Queste policy rappresentano un passo importante per garantire che l'adozione di GenAI nel contesto accademico sia sicura e responsabile. Con il rapido sviluppo dell'IA, sarà cruciale monitorare e aggiornare costantemente tali linee guida, bilanciando innovazione e integrità accademica, per assicurare che la tecnologia continui a essere un supporto senza compromettere la qualità del processo scientifico.

CAPITOLO 3

Proposta

3.1 Metodo di ricerca

L'obiettivo principale di questo lavoro è addestrare un LLM per generare revisioni scientifiche in modo autonomo, sfruttando la tecnica del Fine-Tuning [54]. Partendo da un dataset di articoli scientifici e relative revisioni, il modello viene adattato per acquisire le competenze necessarie a valutare criticamente i paper. Attraverso il Fine-Tuning, l'LLM, che ha già un'ampia conoscenza di base grazie al pre-addestramento cui è stato sottoposto, è esposto a un ampio corpus di contenuti accademici e alle revisioni corrispondenti, sviluppando così una comprensione delle strutture, dei criteri di valutazione e delle convenzioni stilistiche tipiche delle Peer Review [53].

Il fine ultimo è fare in modo che, dato un articolo scientifico in input, il modello sia in grado di generare una revisione completa e coerente come output, imitando il processo di valutazione critica svolto dai revisori umani. Il processo di addestramento e inferenza si articola quindi in due fasi principali: la preparazione e l'elaborazione del dataset per il Fine-Tuning del modello, e l'utilizzo del modello addestrato per generare revisioni, valutandone poi la coerenza e la qualità rispetto a revisioni umane. Per ottenere una valutazione più completa e approfondita delle prestazioni, sono stati selezionati, addestrati e confrontati diversi modelli di LLM, ognuno configurato

con parametri e impostazioni differenti. Questo approccio ha permesso di analizzare in dettaglio le capacità dei vari modelli, al fine di identificare quello in grado di generare le revisioni più coerenti e accurate.

3.1.1 Preparazione dei dati

I dati utilizzati per questa ricerca, costituiti da coppie di articoli scientifici e relative revisioni, sono stati forniti dal laboratorio di ricerca *SeSa*, con cui è stata svolta la tesi. Le revisioni sono state redatte dai membri del laboratorio come parte di un'attività interna di Peer Review, condotta sugli articoli prodotti dallo stesso *SeSa Lab* prima della sottomissione a riviste scientifiche, garantendo così una coerenza del contenuto critico. È importante sottolineare che queste revisioni non violano le policy sull'uso di GenAI (2.5), poiché sono state realizzate prima dell'avvento di tali tecnologie e si basano esclusivamente su artefatti interni al laboratorio. Pertanto, questo lavoro rappresenta una simulazione realistica di un processo di Peer Review interno, rispettando i principi di integrità e conformità etica.

Per organizzare il dataset, è stato sviluppato un codice in Python per unire ogni articolo alla sua rispettiva revisione, formando un dataset strutturato in formato JSON, in cui ogni entry è composta dai seguenti campi: `researcher`, `review_id`, `paper` e `review`. In particolare, i campi `researcher` e `review_id` sono stati inseriti per permettere un'eventuale specializzazione del modello in sviluppi futuri. Il campo `researcher` permetterebbe di tracciare l'autore della revisione, rendendo possibile adattare il modello allo stile specifico di un singolo ricercatore nel caso in cui siano disponibili molte revisioni dello stesso autore, mentre `review_id` offre un codice univoco per ciascuna revisione, facilitando l'identificazione e la gestione delle revisioni individuali.

Per la suddivisione del dataset, è stato adottato il principio di Pareto, comunemente utilizzato in ambito statistico e di Machine Learning, che prevede di assegnare l'80% dei dati al dataset di training e il restante 20% al dataset di test. Il dataset finale contiene un totale di 67 entry, suddivise in 53 entry per il training e 14 per il test. Questa divisione ottimizza il volume di dati disponibili per l'apprendimento del modello, garantendo al contempo un set di dati separato per la valutazione delle

prestazioni su input non visti, e quindi una stima più affidabile della capacità di generalizzazione del modello [60].

In base alle ricerche di *Raffel* et al. [6], la struttura del dataset di coppie paper-review è stata scelta per sfruttare al meglio le tecniche moderne di Transfer Learning, in cui modelli di linguaggio vengono pre-addestrati su grandi quantità di dati non etichettati e successivamente adattati a compiti specifici tramite il Fine-Tuning. *Raffel* et al. sottolineano che il pre-addestramento su dati non etichettati tramite apprendimento non supervisionato ha permesso di ottenere risultati all'avanguardia in diversi benchmark NLP, dimostrando che l'esposizione ad ampie quantità di dati testuali non etichettati consente al modello di acquisire conoscenze utili in una vasta gamma di compiti. Questo approccio, che non richiede etichette per i dati, permette al modello di apprendere rappresentazioni linguistiche e strutturali generali.

3.2 Metodologia

In questa sezione vengono descritti i tre modelli principali impiegati per affrontare il problema della generazione di revisioni scientifiche: *T5 Base*, *Flan-T5 XXL* e *LED Large*. Basati sull'architettura Transformer (2.4.1), questi modelli sono stati selezionati per le loro capacità avanzate di gestione dei compiti di generazione di linguaggio naturale. Ciascuno di essi presenta caratteristiche distintive che influenzano le prestazioni in scenari specifici.

T5 Base Il primo modello adottato è stato T5 Base, un framework *Text-to-Text* introdotto da *Raffel* et al. [6]. Con 220 milioni di parametri, T5 Base rappresenta una scelta versatile e leggera per compiti di elaborazione del linguaggio naturale, rendendolo ideale per contesti con risorse computazionali limitate. Nonostante la sua flessibilità, T5 Base presenta alcune limitazioni significative. La lunghezza massima degli input è limitata a 512 token, un vincolo che può essere problematico per la generazione di revisioni basate su articoli scientifici, che eccedono questa lunghezza. Inoltre, la capacità del modello di produrre testi altamente dettagliati e contestuali è inferiore rispetto a modelli più avanzati. Per questi

motivi, T5 Base è stato principalmente utilizzato come punto di partenza per le prime fasi del lavoro.

Flan-T5 XXL Grazie a tecniche come LoRA e la Quantizzazione, è stato possibile effettuare dei test con Flan-T5 XXL [61], una versione avanzata di T5 con 11 miliardi di parametri. Questo modello rappresenta un enorme passo avanti rispetto a T5, grazie alla sua capacità di generare testi altamente coerenti, dettagliati e stilisticamente raffinati. Flan-T5 XXL si distingue nettamente da T5 Base e LED Large per le seguenti ragioni:

- Comprensione contestuale avanzata: Grazie al suo pre-addestramento su una gamma estesa di compiti e dati, Flan-T5 XXL eccelle nell'identificare sfumature linguistiche e generare risposte tecniche, formali e complesse.
- Capacità di generalizzazione: Il modello dimostra una straordinaria versatilità nell'adattarsi a contesti diversi, fornendo output di qualità significativamente superiore rispetto a T5 Base e LED Large.
- Dettaglio, coerenza e capacità critica: Flan-T5 XXL riesce a produrre revisioni stilisticamente sofisticate, con un livello di coerenza e sintassi che si avvicina al contributo umano. Durante i test, è stato inoltre osservato che il modello esprimeva una maggiore capacità critica, dimostrando di saper formulare osservazioni approfondite e contestuali rispetto al contenuto analizzato.

Tuttavia, il principale limite di *Flan-T5 XXL* è la lunghezza massima degli input, fissata a 512 o 1024 token. Sebbene sia enormemente superiore a T5 Base e LED Large in termini di capacità generativa, questo limite impone l'adozione di tecniche di suddivisione per gestire articoli scientifici completi. Se non fosse per questa restrizione, Flan-T5 XXL rappresenterebbe un ottimo candidato per questo tipo di task.

LED Large Per affrontare il limite di lunghezza degli input di Flan-T5 XXL e T5 Base, è stato adottato LED Large, un modello progettato specificamente per gestire sequenze di grandi dimensioni, fino a 16.384 token [8]. LED Large utilizza un meccanismo di attenzione scalabile che combina attenzione locale e globale,

riducendo il costo computazionale per testi lunghi. Sebbene LED Large sia altamente efficiente nella gestione di input lunghi, il modello presenta una grande limitazioni rispetto a Flan-T5 XXL, ovvero il dettaglio e la qualità del testo: mentre LED Large è eccellente nel mantenere una visione globale del contenuto, le sue capacità di generare testi raffinati e stilisticamente sofisticati sono inferiori rispetto a Flan-T5 XXL. Nonostante queste limitazioni, la capacità di LED Large di elaborare documenti lunghi lo rende indispensabile per scenari in cui i vincoli di lunghezza degli input di Flan-T5 XXL non possono essere superati.

In sintesi, l'utilizzo di T5 Base, Flan-T5 XXL e LED Large ha permesso di valutare diversi modelli di linguaggio rispetto al compito della generazione di revisioni scientifiche. Ogni modello presenta vantaggi e limitazioni specifici: T5 Base si distingue per la leggerezza computazionale ma soffre di vincoli di lunghezza degli input e degli output [6]; Flan-T5 XXL offre capacità generative avanzate e un'elevata qualità dei testi, ma il limite di lunghezza dei token e le elevate risorse computazionali necessarie ne riducono l'applicabilità per documenti lunghi [61]; infine, LED Large eccelle nella gestione di input estesi grazie al suo meccanismo di attenzione scalabile, ma sacrifica in parte il dettaglio e la raffinatezza degli output rispetto a Flan-T5 XXL [8].

Durante le fasi di addestramento e inferenza, è stato necessario impostare l'opzione `truncation=True` per ciascun modello al fine di evitare di eccedere i limiti di memoria disponibili, sia del modello stesso che della GPU utilizzata. Sebbene questa configurazione si sia rivelata essenziale per garantire il corretto svolgimento dei processi, ha spesso limitato la capacità del modello di elaborare interamente gli input più lunghi. Questo ha comportato il troncamento di informazioni potenzialmente essenziali, ostacolando la generazione di revisioni complete e coerenti. Questa analisi sottolinea la necessità di sviluppare soluzioni più efficaci per la gestione di testi lunghi, puntando a bilanciare in modo ottimale la qualità degli output generati con i vincoli imposti dalle limitazioni hardware e dalle architetture dei modelli di linguaggio attualmente disponibili.

3.2.1 Configurazioni e fase di training

L'intero processo di sviluppo, dal training al testing fino all'inferenza, è stato implementato utilizzando i pacchetti *Transformers* di *Hugging Face* [62] e *Torch* [63].

- Transformers di Hugging Face: Questa libreria fornisce un'ampia gamma di modelli pre-addestrati e strumenti per il Fine-Tuning. In questo lavoro, è stata utilizzata per accedere alle implementazioni di T5 e LED, semplificando il caricamento del modello, la configurazione del tokenizer, e l'esecuzione di training e inferenza. La flessibilità di Transformers ha permesso di adattare le pipeline alle esigenze specifiche del progetto.

- Torch: PyTorch, un framework di Deep Learning open source, è stato utilizzato per ottimizzare i modelli, gestire le operazioni tensoriali e sfruttare le capacità della GPU durante il training, il testing e l'inferenza.

- Ecosistema Hugging Face: Oltre alla libreria Transformers, l'ecosistema Hugging Face include strumenti utili per la gestione del dataset, la tokenizzazione e la valutazione delle prestazioni (e.g., il modulo `datasets` è stato utilizzato per il caricamento e la manipolazione del dataset).

L'utilizzo combinato di questi strumenti ha consentito di sviluppare un processo modulare ed efficiente, in grado di adattarsi ai requisiti di ciascun modello e ai limiti hardware disponibili. In particolare, la libreria Transformers di Hugging Face si è dimostrata ottimale per il lavoro svolto, grazie alla capacità di fornire implementazioni preconfigurate dei modelli utilizzati, semplificando l'adattamento ai diversi task di Fine-Tuning. Questa flessibilità ha permesso un'integrazione senza interruzioni tra i modelli, agevolando il caricamento, la configurazione e il riutilizzo per compiti specifici. Un ulteriore punto di forza della libreria è la sua modularità, che consente di combinare facilmente tokenizer, dataset e configurazioni senza richiedere interventi significativi, migliorando così l'efficienza del processo di sviluppo. Questo approccio ha reso possibile la sperimentazione con diversi modelli in modo più rapido e flessibile.

Preparazione dei dataset

La prima fase del lavoro è stata comune a tutti i modelli e ha riguardato la preparazione dei dataset e la loro trasformazione in un formato compatibile con le librerie Hugging Face Transformers. I dataset di training e test, salvati in formato JSON, sono stati caricati e convertiti in oggetti `DatasetDict`, che rappresentano la struttura standard per la gestione dei dati in Hugging Face. Il codice riportato di seguito mostra i passaggi fondamentali di questa fase:

```
train_path = os.path.abspath("../../data/processed/train_dataset.json")
test_path = os.path.abspath("../../data/processed/test_dataset.json")

# Load the training and test datasets
train_data, test_data = load_datasets(train_path, test_path)

train_dataset = convert_to_dataset(train_data)
test_dataset = convert_to_dataset(test_data)

datasets = DatasetDict({"train": train_dataset, "test": test_dataset})

# Print dataset features and sizes
print(colored(f"Train Dataset Features: {datasets['train'].features}", "
    cyan"))
print(colored(f"Test Dataset Features: {datasets['test'].features}", "
    cyan"))
print(colored(f"Train Dataset Size: {len(datasets['train'])}", "cyan"))
print(colored(f"Test Dataset Size: {len(datasets['test'])}\n", "cyan"))
```

Listing 3.1: Caricamento e preparazione dei dataset

I file JSON contenenti il dataset di training e quello di test sono stati caricati attraverso l'utilizzo della funzione `load_datasets`, che si occupa del parsing e della validazione dei dati, garantendo che siano pronti per le fasi successive di elaborazione. Una volta caricati, i dati sono stati trasformati in oggetti `Dataset` grazie alla funzione `convert_to_dataset`, specificamente progettata per renderli compatibili con le API della libreria Hugging Face. Questo passaggio ha assicurato una gestione efficiente e uniforme dei dati all'interno del framework. Successivamente, i dataset di training e di test sono stati organizzati in un unico oggetto `DatasetDict`, che

ne semplifica l'accesso e la manipolazione durante le fasi di training e validazione. Infine, è stata implementata una stampa delle caratteristiche (feature) e delle dimensioni di ciascun dataset, per verificare visivamente la correttezza della preparazione iniziale e garantire che i dati fossero pronti per essere processati dal modello. L'implementazione delle due funzioni `load_datasets` e `convert_to_dataset` è qui presentata.

```
# Load the training and validation datasets
def load_datasets(train_path, test_path):
    with open(train_path, "r", encoding="utf-8") as f:
        train_data = json.load(f)
    with open(test_path, "r", encoding='utf-8') as f:
        test_data = json.load(f)

    return train_data, test_data

# Convert the dataset to Datasets library format
def convert_to_dataset(data):
    flattened_data = []
    for item in data:
        flattened_data.append({
            "paper": item["paper"],
            "review": item["review"]
        })
    return Dataset.from_pandas(pd.DataFrame(flattened_data))
```

Listing 3.2: Funzioni di preparazione dei dataset

La funzione `load_datasets` è responsabile del caricamento dei file JSON contenenti i dataset di training e di test, aprendo ciascun file in modalità di lettura con codifica UTF-8. Il contenuto viene quindi deserializzato in oggetti Python utilizzando la libreria `json`, consentendo di estrarre e organizzare i dati sotto forma di liste di dizionari. Per garantire la compatibilità con Hugging Face Datasets, la funzione `convert_to_dataset` si occupa di riorganizzare i dati, trasformando ogni elemento in un dizionario strutturato con i campi `paper` e `review`, che rappresentano rispettivamente l'articolo scientifico e la revisione associata. Infine, i dati riorganizzati vengono convertiti in un oggetto `Dataset` tramite un DataFrame Pandas, che li

rende pronti per essere utilizzati nelle fasi successive del processo.

Caricamento dei modelli e dei tokenizer

Il caricamento dei modelli e dei rispettivi tokenizer è stato effettuato utilizzando le API di Hugging Face Transformers. Ogni modello richiede configurazioni specifiche in base alla propria architettura e alle tecniche di ottimizzazione applicate. Di seguito vengono illustrate le configurazioni adottate per ciascun modello: T5 Base, LED Large e Flan-T5 XXL.

Per T5 Base, il modello e il tokenizer sono stati caricati utilizzando le rispettive funzioni dedicate. La funzione `preprocess_function` è stata applicata ai dataset attraverso la funzione `map()` di Hugging Face Datasets, per generare un dataset tokenizzato compatibile con il modello.

```
# Load the T5 model and tokenizer
model_name = "t5-base"
tokenizer = T5Tokenizer.from_pretrained(model_name, legacy=False)
model = T5ForConditionalGeneration.from_pretrained(model_name)

tokenized_datasets = datasets.map(preprocess_function, batched=True)
```

Listing 3.3: Caricamento di T5 Base e rispettivo tokenizer

Il caricamento di Flan-T5 XXL ha richiesto configurazioni specifiche per ottimizzare l'uso delle risorse hardware. È stata utilizzata una versione sharded del modello con precisione FP16, la suddivisione in shard e l'uso della precisione FP16 hanno ridotto ulteriormente il consumo di memoria. Inoltre, è stata applicata la quantizzazione a 8-bit tramite la classe `BitsAndBytesConfig`, e il modello è stato adattato al task con il framework LoRA, limitando il numero di parametri addestrabili per migliorare l'efficienza del training. Anche in questo caso, la tokenizzazione dei dati è stata gestita dalla funzione `preprocess_function`, che ha svolto un ruolo essenziale nel garantire la compatibilità e l'efficienza del processo. Per quanto riguarda LoRA, la configurazione adottata prevedeva `r=16` e `lora_alpha=32`, parametri scelti attentamente sulla base di osservazioni empiriche condotte durante vari esperimenti di addestramento. Un ulteriore incremento di questi valori avrebbe comportato un consumo di memoria significativamente più elevato, portando a stalli durante il

training su GPU e tempi di addestramento estremamente prolungati, in alcuni casi superiori alle 24 ore, rendendo il processo inefficiente e impraticabile.

```
# Load the T5 model and tokenizer
model_name = "philschmid/flan-t5-xxl-sharded-fp16"
quantization_config = BitsAndBytesConfig(
    load_in_8bit=True,
)
device_map = {'' : torch.cuda.current_device()}
model = AutoModelForSeq2SeqLM.from_pretrained(
    model_name,
    quantization_config=quantization_config,
    device_map=device_map
)
tokenizer = AutoTokenizer.from_pretrained(model_name)
# Apply Lora
lora_config = LoraConfig(
    task_type=TaskType.SEQ_2_SEQ_LM,
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    target_modules=["q", "v"],
    bias="none"
)
# Prepare int-8 model for training
model = prepare_model_for_kbit_training(model)
# Add Lora adaptor
model = get_peft_model(model, lora_config)
model.print_trainable_parameters()

tokenized_datasets = datasets.map(preprocess_function, batched=True)
```

Listing 3.4: Caricamento di Flan-T5 XXL e rispettivo tokenizer

Per LED Large, il caricamento del modello e del tokenizer segue un processo sostanzialmente analogo a quello previsto per T5 Base, con l'unica differenza che sono state impiegate le classi specificamente progettate per LED. Anche in questo caso, come per T5 Base, grazie al numero di parametri relativamente gestibile rispetto a modelli più avanzati come Flan-T5 XXL, non è stato necessario ricorrere a tecniche

aggiuntive, quali la Quantizzazione o l'integrazione di LoRA, semplificando così il processo di configurazione e addestramento.

```
# Load the model and tokenizer
model_name = "allenai/led-large-16384"
model = LEDForConditionalGeneration.from_pretrained(model_name,).to("cuda")
tokenizer = LEDTokenizer.from_pretrained(model_name)

tokenized_datasets = datasets.map(preprocess_function, batched=True)
```

Listing 3.5: Caricamento di LED Large e rispettivo tokenizer

Le funzioni `preprocess_function` sono state definite per ciascun modello al fine di preparare i dati in modo compatibile con i rispettivi tokenizer. Per T5 Base e Flan-T5 XXL, la funzione riceve come input il campo `paper` del dataset e lo tokenizza con una lunghezza massima di 512 token, applicando troncamento e padding per uniformare la lunghezza degli input. Il campo `review` viene tokenizzato separatamente per generare le etichette (`labels`), anch'esse limitate a un massimo di 512 token. I token risultanti vengono inclusi nel dizionario `model_inputs` come `input_ids` e `labels`, rispettivamente.

```
# Tokenize the dataset
def preprocess_function(examples):
    inputs = [ex for ex in examples["paper"]]
    targets = [ex for ex in examples["review"]]
    model_inputs = tokenizer(inputs, max_length=512, truncation=True, padding="max_length")

    # Setup the tokenizer for targets
    labels = tokenizer(text_target=targets, max_length=512, truncation=True, padding="max_length")

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs
```

Listing 3.6: Funzione di preprocessing di T5 Base e Flan-T5 XXL

Per LED Large, la funzione segue un approccio simile ma adattato alle specifiche del modello. Gli input (`paper`) sono tokenizzati fino a una lunghezza massima di 16.384

token, sfruttando la capacità del modello di gestire sequenze lunghe. Le etichette (`review`), invece, sono state impostate a un massimo di 1.024 token, poiché superare questo valore avrebbe ecceduto i limiti di memoria disponibili. Anche in questo caso, vengono applicati troncamento e padding.

Definizione dei parametri di training

Per garantire coerenza nel processo di addestramento, sono stati utilizzati gli stessi parametri di training per tutti e tre i modelli (T5 Base, Flan-T5 XXL e LED Large). Questi parametri sono stati definiti con l'obiettivo di bilanciare efficacemente la qualità del training e l'efficienza computazionale, tenendo conto delle risorse hardware disponibili. I parametri impostati sono i seguenti:

```
# Define the training arguments
training_args = TrainingArguments(
    output_dir="../../results", # The output directory
    eval_strategy="epoch", # Evaluation is done at the end of each epoch
    num_train_epochs=5, # Number of training epochs
    learning_rate=1e-4, # Learning rate
    per_device_train_batch_size=1, # Batch size for training
    per_device_eval_batch_size=1, # Batch size for evaluation
    weight_decay=0.01, # Weight decay to avoid overfitting
    save_total_limit=5, # Limit the total number of checkpoints
    save_strategy="epoch", # Save a checkpoint at the end of each epoch
    logging_steps=10, # Log metrics every 10 steps
    load_best_model_at_end=True, # Load the best model when training ends
    metric_for_best_model="eval_loss", # Use loss to determine the best
    model
    greater_is_better=False, # A lower loss is better
    dataloader_drop_last=True, # Consider last batch even if it's smaller
)
```

Listing 3.7: Parametri di training

La configurazione dei parametri chiave, come il `weight_decay`, il `learning_rate` e il `num_train_epoch`, è stata determinata tramite osservazioni empiriche derivate da numerosi esperimenti di addestramento. Per i modelli LED Large e T5 Base, sono stati condotti circa 20 training differenti, consentendo di testare varie combinazioni

di parametri e ottimizzare le prestazioni. Per Flan-T5 XXL, invece, il numero di esperimenti è stato limitato a circa 10, poiché i tempi di training erano significativamente più lunghi a causa della complessità del modello. Inoltre, per Flan-T5 XXL, il numero massimo di epoche è stato ridotto a 5, mentre per gli altri modelli si è potuto raggiungere un numero di 10 epoche. Questi test hanno permesso di individuare valori che bilanciano l'efficacia dell'apprendimento con il rischio di overfitting. Al contrario, la scelta del `batch_size` è stata influenzata principalmente dai vincoli hardware e dalla complessità del modello utilizzato, con l'obiettivo di ottimizzare il consumo di memoria GPU. La metrica principale utilizzata per valutare le prestazioni durante l'addestramento è stata la `eval_loss`, considerata il parametro chiave per monitorare l'efficacia del processo di ottimizzazione. Una `eval_loss` più bassa indica una migliore capacità del modello di apprendere i pattern presenti nei dati senza andare in overfitting. Il modello migliore è stato selezionato principalmente sulla base della `eval_loss` più bassa, calcolata durante la fase di validazione, in quanto questo valore è stato ritenuto il fattore più influente nel determinare la qualità generale del modello. Questa scelta ha garantito che il modello finale fosse quello con la migliore generalizzazione ai dati di test.

Optimizer, Scheduler ed Early Stopping

Dopo aver effettuato numerosi esperimenti sui vari modelli è stato scelto l'Optimizer *AdamW* [64], noto per la sua capacità di gestire efficacemente il decadimento dei pesi, garantendo stabilità durante il training. In parallelo, è stato sperimentato *Adafactor* [65], come raccomandato da *Raffel* et al. [6], e ulteriormente supportato da *Shazeer* e *Stern*, che evidenziano il suo vantaggio nell'ottimizzazione della memoria, un aspetto cruciale per modelli di grandi dimensioni. Tuttavia, i test effettuati hanno mostrato che AdamW ha prodotto risultati superiori in termini di convergenza e qualità complessiva del modello, portando alla sua adozione come soluzione definitiva.

Anche per lo Scheduler, è stato effettuato un confronto, in questo caso tra l'approccio lineare e quello basato sul coseno. Lo Scheduler lineare si è dimostrato più stabile e affidabile, fornendo risultati più consistenti e riducendo la variabilità delle

prestazioni tra gli esperimenti. Per evitare un training eccessivo e ottimizzare i tempi, è stato implementato un criterio di Early Stopping che ferma il training se la metrica di riferimento (`eval_loss`) non migliorava ulteriormente per 3 epoche. Queste configurazioni, che includono l'uso di AdamW, lo Scheduler lineare e l'Early Stopping, sono state utilizzate in modo equivalente per tutti i modelli.

```
# Define the optimizer and scheduler
optimizer = optim.AdamW(model.parameters(), lr=training_args.
    learning_rate)
total_steps = len(tokenized_datasets["train"]) * training_args.
    num_train_epochs
scheduler = get_linear_schedule_with_warmup(optimizer, num_warmup_steps
    =0, num_training_steps=total_steps)

# Define the early stopping callback
early_stopping_callback = EarlyStoppingCallback(early_stopping_patience
    =3)
```

Listing 3.8: Optimizer, Scheduler ed Early Stopping

Esecuzione del training

Dopo aver completato la configurazione dei modelli, degli Optimizer, degli Scheduler e degli altri parametri necessari, è stato avviato il processo di training per ciascun modello. Prima dell'inizio del training, sono state caricate le metriche necessarie ed è stato definito il `Trainer`. Durante il training, le prestazioni del modello sono state valutate ad ogni epoca calcolando le metriche di valutazione, che confrontano il testo generato dal modello con il testo di riferimento presente nel dataset di test. Tali metriche verranno descritte in dettaglio nella sezione dedicata alla validazione.

3.2.2 Valutazione e testing

La valutazione delle performance dei modelli di NLP è fondamentale per determinare la loro efficacia. Le metriche di valutazione comunemente utilizzate nel NLP si concentrano su aspetti come la somiglianza tra il testo generato e quello di riferimento, la capacità del modello di mantenere la coerenza semantica, e l'accura-

tezza nel catturare le relazioni linguistiche e contestuali. Le metriche utilizzate per valutare ciascun modello sono state *BLEU*, *ROUGE*, *METEOR* e *BERTScore*, ciascuna delle quali offre una prospettiva unica sulla qualità del testo. Si noti che la Funzione di Perdita è cruciale per monitorare l'andamento dell'apprendimento del modello durante l'addestramento. Tuttavia, le metriche di valutazione sono adatte per fornire un'indicazione complementare per la valutazione del modello. Di seguito viene fornita una descrizione delle metriche menzionate.

BLEU La metrica BLEU (Bilingual Evaluation Understudy) è una delle metriche più utilizzate per la valutazione automatica della traduzione e del testo generato, introdotta da *Papineni* et al. nel 2002 [66]. BLEU valuta la qualità di un testo generato confrontandolo con uno o più testi di riferimento, basandosi sulla similarità tra sequenze di parole (n-grammi) nel testo generato e in quelli di riferimento. Questa metrica assegna un punteggio in base alla proporzione di n-grammi che il testo generato condivide con i riferimenti, dove un punteggio più alto indica una maggiore somiglianza. BLEU calcola la precisione per n-grammi di diverse lunghezze e applica una penalizzazione per evitare che frasi troppo corte ricevano punteggi elevati. La formula principale per BLEU è:

$$BLEU = BP \cdot exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3.2.1}$$

Dove:

$BP =$ *Brevity Penalty*, un fattore di penalizzazione per le frasi corte

$p_n =$ Precisione per ciascun n-gramma

$w_n =$ Pesi associati a ciascun n-gramma

La precisione per n-grammi (p_n) è calcolata come il rapporto tra il numero di n-grammi del testo generato che coincidono con gli n-grammi del riferimento e il numero totale di n-grammi nel testo generato. La Brevity Penalty (BP) è introdotta per evitare che testi generati troppo brevi ottengano un punteggio elevato semplicemente perché condividono pochi n-grammi con il riferimento. Viene calcolata come:

$$BP = \begin{cases} 1 & \text{se } c > r \\ e^{(1-r/c)} & \text{se } c \leq r \end{cases} \tag{3.2.2}$$

Dove:

$c =$ Lunghezza del testo generato

$r =$ Lunghezza del testo di riferimento

Sebbene BLEU sia ampiamente utilizzato, presenta alcune limitazioni. In particolare, questa metrica non cattura la similarità semantica o il significato contestuale, concentrandosi esclusivamente sulla corrispondenza di n-grammi. Inoltre, BLEU può penalizzare testi generati validi che utilizzano parafrasi o sinonimi, in quanto valuta solo la precisione degli n-grammi rispetto ai riferimenti senza considerare variazioni linguistiche.

ROUGE La metrica ROUGE (Recall-Oriented Understudy for Gisting Evaluation) è stata introdotta nel 2004 da *Chin-Yew Lin* [67]. A differenza di BLEU, che si concentra principalmente sulla precisione degli n-grammi, ROUGE misura la qualità del testo generato confrontando l'overlap di unità come n-grammi, sequenze di parole e coppie di parole tra il testo generato e i testi di riferimento. Questa caratteristica la rende particolarmente adatta per compiti come il riassunto, dove è importante catturare il contenuto informativo completo. ROUGE include diverse varianti, ognuna delle quali misura aspetti differenti della similarità tra il testo generato e i testi di riferimento.

ROUGE-N valuta l'overlap di n-grammi tra il testo generato e quello di riferimento, calcolando il richiamo degli n-grammi condivisi. La formula di ROUGE-N è:

$$ROUGE - N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{3.2.3}$$

Dove n sta per il numero di n-grammi, $Count_{match}(gram_n)$ è il numero di n-grammi condivisi tra il testo generato e il riferimento, e $Count(gram_n)$ è il numero totale di n-grammi nel riferimento. ROUGE-1 e ROUGE-2 sono le versioni più comuni, calcolando rispettivamente l'overlap di unigrammi e bigrammi.

ROUGE-L si basa invece sulla *Longest Common Subsequence (LCS)*, che rappresenta la lunghezza della sottosequenza comune più lunga tra il testo generato e il riferimento. Per ROUGE-L, il Recall (Rlcs) è calcolato dividendo la lunghezza della LCS tra il testo generato e il testo di riferimento per la lunghezza totale del testo di riferimento. La Precision (Plcs) è calcolata in modo simile, ma divide per la lunghezza del testo generato. L'F-measure (Flcs) viene calcolata combinando Recall e Precision con un fattore di bilanciamento β.

$$Rlcs = \frac{LCS(X,Y)}{m} \tag{3.2.4}$$

$$Plcs = \frac{LCS(X,Y)}{n} \tag{3.2.5}$$

$$Flcs = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs}+\beta^2 P_{lcs}} \tag{3.2.6}$$

Dove $LCS(X,Y)$ rappresenta la lunghezza della LCS tra il testo generato Y e il testo di riferimento X, m è la lunghezza di X, e n è la lunghezza di Y. Questo approccio consente a ROUGE-L di valutare la struttura sequenziale del contenuto, misurando quanto accuratamente il testo generato segue l'ordine del testo di riferimento.

ROUGE-W è una variante di ROUGE-L che introduce un peso per le sottosequenze più lunghe. A differenza di ROUGE-L, che considera solo la lunghezza della sottosequenza comune più lunga (LCS), ROUGE-W assegna un peso maggiore alle sottosequenze più lunghe, permettendo così di penalizzare maggiormente le discontinuità. La metrica considera la continuità del match tra n-grammi, il che la rende particolarmente utile in compiti dove non solo il contenuto, ma anche la sequenzialità è importante. Il peso aggiuntivo applicato ai match continui rende ROUGE-W più sensibile alle interruzioni nella corrispondenza tra le frasi del testo generato e quelle di riferimento.

Infine ROUGE-S misura gli "skip-bigram" condivisi tra il testo generato e il riferimento, dove uno skip-bigram è una coppia di parole che appaiono nello stesso ordine ma non necessariamente in modo contiguo. Questa metrica cattura relazioni tra parole non adiacenti, riflettendo la flessibilità delle strutture linguistiche naturali e rendendola utile quando il significato può essere mantenuto con variazioni nell'ordine delle parole. ROUGE-SU, una variante di

ROUGE-S, include anche i singoli unigrammi, offrendo una valutazione più completa che tiene conto sia delle connessioni tra parole non adiacenti sia della presenza di parole isolate significative.

In sintesi ROUGE è una metrica efficace per valutare il contenuto informativo e la somiglianza strutturale, specialmente nei compiti di riassunto. Tuttavia, non riesce a cogliere pienamente la varietà espressiva e le sfumature linguistiche, penalizzando parafrasi o variazioni sintattiche che mantengono comunque il significato originale.

METEOR La metrica METEOR (Metric for Evaluation of Translation with Explicit ORdering), proposta da *Banerjee* e *Lavie* nel 2005 [68], è stata sviluppata per superare alcune delle limitazioni di BLEU nella valutazione automatica della traduzione e del testo generato. METEOR si basa su un processo di allineamento tra il testo generato e quello di riferimento, cercando di mappare gli unigrammi in base a criteri di corrispondenza che includono:

- Corrispondenza esatta tra parole identiche nel testo generato e nel riferimento.
- Corrispondenza per radice (stem): mappature tra parole che condividono la stessa radice (e.g., "run" e "running").
- Corrispondenza sinonimica: associazioni di parole con significati simili, utilizzando un dizionario di sinonimi.

Dopo aver calcolato le corrispondenze, METEOR utilizza le misure di Precision e Recall per valutare la qualità del testo generato. Il valore di Precision è dato dal rapporto tra il numero di unigrammi mappati nel testo generato e il numero totale di unigrammi nel testo generato, mentre il Recall rappresenta il rapporto tra il numero di unigrammi mappati e il numero totale di unigrammi nel riferimento. Questi due valori sono combinati tramite la media armonica:

$$Fmean = \frac{10PR}{R + 9P} \tag{3.2.7}$$

Per evitare che i testi frammentati ottengano un punteggio elevato, METEOR introduce una Penalità di Frammentazione (*Penalty*), che misura la continuità

delle corrispondenze nel testo generato rispetto al riferimento. È calcolata come:

$$Penalty = \gamma \cdot \left(\frac{\#chunks}{\#unigrams\_matched} \right)^{\beta} \tag{3.2.8}$$

Dove *#chunks* è il numero di segmenti discontinui di unigrammi corrispondenti tra il testo generato e il testo di riferimento e *#unigrams_matched* rappresenta il numero totale di unigrammi mappati. I parametri γ e β controllano l'intensità della penalizzazione per riflettere l'importanza della continuità nelle corrispondenze.

Inoltre, il calcolo del punteggio METEOR è dato da:

$$METEOR = Fmean \cdot (1 - Penalty) \tag{3.2.9}$$

Questa formula considera sia la somiglianza tra il testo generato e il riferimento, sia la continuità delle corrispondenze. Grazie alla capacità di tenere conto di sinonimi e radici, METEOR si è dimostrato efficace nel riflettere la qualità percepita nei giudizi umani.

BERTScore La metrica BERTScore è una metrica avanzata che sfrutta i modelli di embedding contestuali, come BERT, per valutare la qualità del testo generato. Introdotto da *Zhang* et al. nel 2019 [69], BERTScore confronta ogni token del testo generato con i token del testo di riferimento utilizzando la similarità del coseno tra i vettori di embedding, superando le limitazioni delle metriche tradizionali basate su confronti diretti di parole o risorse lessicali. Per calcolare BERTScore, si usano Precision, Recall e F-score, con le seguenti formule:

$$P_{BERT} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^\top \hat{x}_j \tag{3.2.10}$$

$$R_{BERT} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^\top \hat{x}_j \tag{3.2.11}$$

$$F_{BERT} = 2\frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}} \tag{3.2.12}$$

Dove $\hat{x}$ rappresenta il set di token del testo di riferimento e x rappresenta il set di token del testo generato. Il termine $x_i^\top \hat{x}_j$ nella formula indica la similarità del coseno tra i vettori di embedding dei token x_i e $\hat{x}_j$, rispettivamente nel testo generato e in quello di riferimento.

Rispetto a METEOR, che utilizza WordNet per riconoscere sinonimi e variazioni lessicali, BERTScore beneficia del contesto dei token all'interno della frase, catturando relazioni semantiche più complesse. Questo approccio permette a BERTScore di essere più robusto in compiti che richiedono una valutazione ancora più precisa del significato e della coerenza semantica.

Le metriche di valutazione sono state calcolate confrontando le revisioni generate dal modello con quelle di riferimento presenti nel dataset di test, utilizzando la funzione `compute_metrics`. In questa funzione, le predizioni del modello e le etichette di riferimento vengono decodificate tramite il tokenizer, con la rimozione dei token speciali per garantire un confronto accurato. Successivamente, il calcolo di ogni metrica è stato eseguito utilizzando le rispettive funzioni dedicate.

```
# Compute the metrics
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions

    # Convert predictions to the correct format if necessary
    if isinstance(preds, tuple):
        preds = preds[0]

    # Convert predictions to numpy array and take the argmax
    preds = np.argmax(preds, axis=-1)

    # Decode the predicted and actual labels
    decoded_preds = tokenizer.batch_decode(preds, skip_special_tokens=True)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)

    # Compute metrics
    bleu_result = bleu.compute(predictions=decoded_preds, references=[[label] for label in decoded_labels])
    meteor_result = meteor.compute(predictions=decoded_preds, references=decoded_labels)
    bertscore_result = bertscore.compute(predictions=decoded_preds, references=decoded_labels, lang="en")
```

```
    rouge_result = rouge.compute(predictions=decoded_preds, references=
    decoded_labels)

    # Return metrics in a readable format
    metrics = {
        'eval_bleu': bleu_result['bleu'],
        'eval_meteor': meteor_result['meteor'],
        'eval_bertscore_precision': np.mean(bertscore_result['precision'
    ]),
        'eval_bertscore_recall': np.mean(bertscore_result['recall']),
        'eval_bertscore_f1': np.mean(bertscore_result['f1']),
        'eval_rouge1': rouge_result['rouge1'],
        'eval_rouge2': rouge_result['rouge2'],
        'eval_rougeL': rouge_result['rougeL'],
        'eval_rougeLsum': rouge_result['rougeLsum'],
    }

    return metrics
```

Listing 3.9: Funzione di calcolo delle metriche

L'uso combinato di diverse metriche ha garantito una valutazione equilibrata, prendendo in considerazione sia aspetti lessicali che semantici. Questo approccio ha permesso di selezionare il modello con le migliori prestazioni, fornendo una base solida per il successivo utilizzo nella generazione di revisioni.

3.2.3 Generazione delle revisioni

Il processo di inferenza per la generazione delle revisioni è stato implementato tramite una funzione `generate_review`, che varia leggermente a seconda del modello utilizzato. Questa funzione accetta come input il testo dell'articolo scientifico, tokenizzandolo con le opzioni appropriate per ciascun modello, e genera la revisione corrispondente sfruttando i metodi di generazione delle librerie di Hugging Face. I parametri utilizzati per la generazione sono stati scelti sulla base di osservazioni empiriche, effettuando diversi esperimenti per ottimizzare il bilanciamento tra coerenza, precisione e formalità delle revisioni generate. Inizialmente, oltre che sperimentare

con valori differenti, sono stati testati approcci alternativi come l'uso combinato di `top_k`, `top_p` e `do_sample` per aumentare la diversità dell'output. Tuttavia, i parametri attuali, hanno dimostrato di produrre revisioni più formali, coerenti e precise.

```
# Generate a review for a given paper
def generate_review(model, tokenizer, paper_text):
    inputs = tokenizer(paper_text, return_tensors="pt", max_length=512,
   truncation=True, padding="max_length")
    input_ids = inputs["input_ids"].to(model.device)
    attention_mask = inputs["attention_mask"].to(model.device)
    outputs = model.generate(
        input_ids,
        attention_mask=attention_mask,
        max_length=512,
        num_beams=5,
        repetition_penalty=1.3,
        no_repeat_ngram_size=3,
        early_stopping=True,
    )
    review = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return review
```

Listing 3.10: Funzione di generazione delle revisioni per T5 Base e Flan-T5 XXL

Per T5 Base e Flan-T5 XXL, è stata utilizzata una lunghezza massima di input e output di 512 token. La tokenizzazione avviene tramite il tokenizer associato al modello, con i parametri `truncation=True` e `padding="max_length"` per garantire che il testo rientri nei limiti imposti dal modello. Gli output sono generati utilizzando il metodo `generate` con i seguenti parametri:

- `num_beams=5`: Utilizza il beam search per migliorare la qualità del testo generato.
- `repetition_penalty=1.3`: Evita ripetizioni indesiderate durante la generazione.
- `no_repeat_ngram_size=3`: Impedisce la ripetizione di sequenze di tre token consecutivi.

- `early_stopping=True`: Interrompe la generazione quando viene raggiunta una probabile conclusione, risparmiando risorse computazionali.

Il risultato generato viene decodificato rimuovendo eventuali token speciali, restituendo una revisione testuale coerente e leggibile.

```
# Generate a review for a given paper
def generate_review(model, tokenizer, paper_text):
    inputs = tokenizer(paper_text, return_tensors="pt", max_length=16384,
    truncation=True, padding="max_length")
    input_ids = inputs["input_ids"].to(model.device)
    attention_mask = inputs["attention_mask"].to(model.device)
    global_attention_mask = torch.zeros_like(attention_mask)
    global_attention_mask[:, 0] = 1
    outputs = model.generate(
        input_ids,
        attention_mask=attention_mask,
        global_attention_mask=global_attention_mask,
        max_length=1024,
        num_beams=5,
        repetition_penalty=1.3,
        no_repeat_ngram_size=3,
        early_stopping=True,
    )
    review = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return review
```

Listing 3.11: Funzione di generazione delle revisioni per LED Large

Per LED Large, il processo è stato adattato per gestire input di lunghezza fino a 16.384 token, sfruttando una delle caratteristiche principali di questo modello. La tokenizzazione è simile, ma con un limite di lunghezza superiore per l'input. Tuttavia, il massimo numero di token per l'output è stato fissato a 1.024, poiché valori superiori richiederebbero risorse computazionali eccedenti i limiti hardware disponibili. Un'ulteriore distinzione è rappresentata dall'uso della `global_attention_mask`, una maschera specifica di LED Large che assegna maggiore attenzione ai primi token dell'input. Questo approccio consente al modello di focalizzarsi su sezioni chiave del documento per produrre una revisione più accurata.

CAPITOLO 4

Conclusione

4.1 Risultati e analisi

In questo capitolo verranno presentati e analizzati i risultati ottenuti durante l'addestramento dei modelli utilizzati, con particolare attenzione alle metriche di valutazione chiave. I modelli esaminati, tra cui *T5 Base*, *Flan-T5 XXL* e *LED Large*, sono stati confrontati per valutarne le prestazioni nella generazione di revisioni scientifiche. Le metriche utilizzate per questa analisi includono BLEU, METEOR, ROUGE e BERTScore, che misurano aspetti fondamentali della qualità del testo generato, come coerenza, accuratezza e aderenza al testo di riferimento (3.2.2).

Il confronto tra i modelli ha consentito di evidenziare i loro punti di forza e debolezza, mostrando come configurazioni specifiche, parametri di training, caratteristiche del dataset e risorse computazionali abbiano influenzato i risultati ottenuti.

Questa analisi comparativa è stata fondamentale per comprendere l'impatto delle scelte progettuali e delle configurazioni sul rendimento complessivo del modello. I risultati saranno presentati tramite un grafico che mostra l'andamento delle metriche, un grafico dell'andamento della Loss e una heatmap, offrendo una visualizzazione chiara e intuitiva delle prestazioni dei modelli rispetto alle metriche. Per ogni modello

verrà mostrato il miglior risultato ottenuto durante l'addestramento, consentendo una valutazione accurata delle prestazioni massime raggiunte. Ogni risultato sarà analizzato singolarmente e successivamente confrontato con gli altri.

4.1.1 Risultati del modello T5 Base

Andamento delle metriche

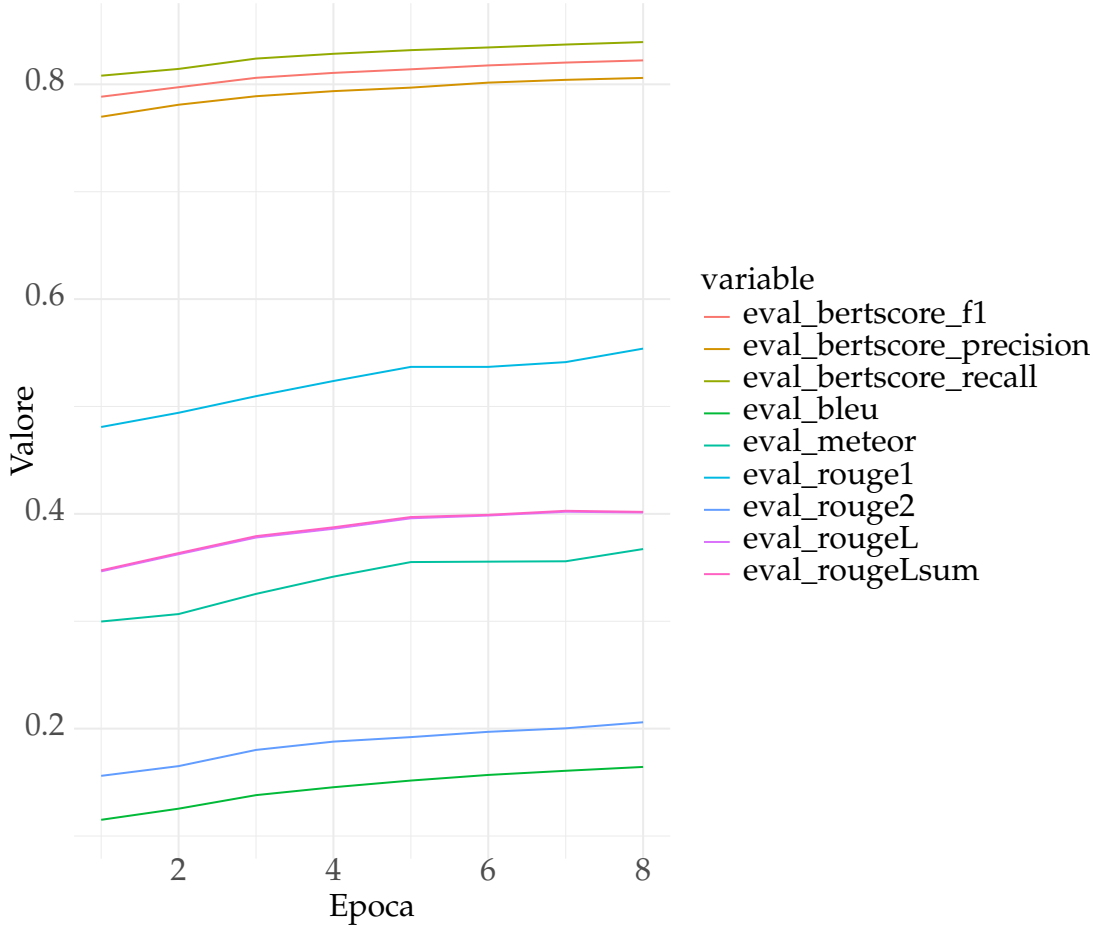

Figura 4.1: Grafico dell'andamento delle metriche (T5 Base)

Il grafico di andamento delle metriche (4.1) mostra l'evoluzione dei valori di diverse metriche di valutazione durante le epoche di addestramento. Dal grafico emerge che tutte le metriche migliorano gradualmente nelle prime epoche, per poi stabilizzarsi con il progredire dell'addestramento, indicando che il modello ha raggiunto un livello di convergenza.

Le metriche BERTScore F1, BERTScore Precision e BERTScore Recall presentano valori elevati già dalle prime epoche e migliorano ulteriormente fino a raggiungere una stabilità nelle epoche conclusive. Questo risultato evidenzia la capacità del modello di mantenere una forte coerenza semantica tra il testo generato e quello di riferimento. Al contrario, la metrica BLEU rimane significativamente più bassa rispetto alle altre, anche nelle epoche finali. Questo potrebbe indicare che il modello fatica a replicare fedelmente le sequenze di n-grammi esatte del testo di riferimento, un comportamento comune nei task di generazione di testo complesso, dove la creatività del modello e la varietà lessicale giocano un ruolo più prominente.

La metrica METEOR, sebbene migliori costantemente durante l'addestramento, si attesta anch'essa su valori relativamente bassi rispetto ad altre metriche. Ciò suggerisce che, pur essendo in grado di catturare alcune corrispondenze lessicali e sinonimiche, il modello potrebbe non eccellere in termini di copertura lessicale rispetto al riferimento.

Le metriche ROUGE-1, ROUGE-2, ROUGE-L e ROUGE-Lsum, che valutano diversi aspetti della sovrapposizione tra il testo generato e quello di riferimento, mostrano un comportamento più favorevole. In particolare, ROUGE-1 e ROUGE-Lsum crescono più rapidamente e si stabilizzano su valori significativi, suggerendo che il modello riesce a catturare bene le strutture generali del testo, mentre ROUGE-2, che misura la corrispondenza di bigrammi, si attesta su valori inferiori.

In sintesi, il grafico conferma che il processo di addestramento del modello T5 Base è stato efficace, con metriche che indicano una buona capacità di generazione testuale, soprattutto in termini di coerenza strutturale e semantica. Tuttavia, i valori relativamente bassi di BLEU e METEOR suggeriscono che il modello potrebbe non eccellere nella riproduzione esatta del testo di riferimento, ma ciò potrebbe riflettere anche la natura più creativa e diversificata delle revisioni generate.

Andamento della Loss

Il grafico dell'andamento della Loss (4.2) mostra una rapida diminuzione nelle prime epoche, segnalando una fase iniziale di apprendimento efficace da parte del modello. La Loss raggiunge un minimo intorno alla quarta epoca, indicando un

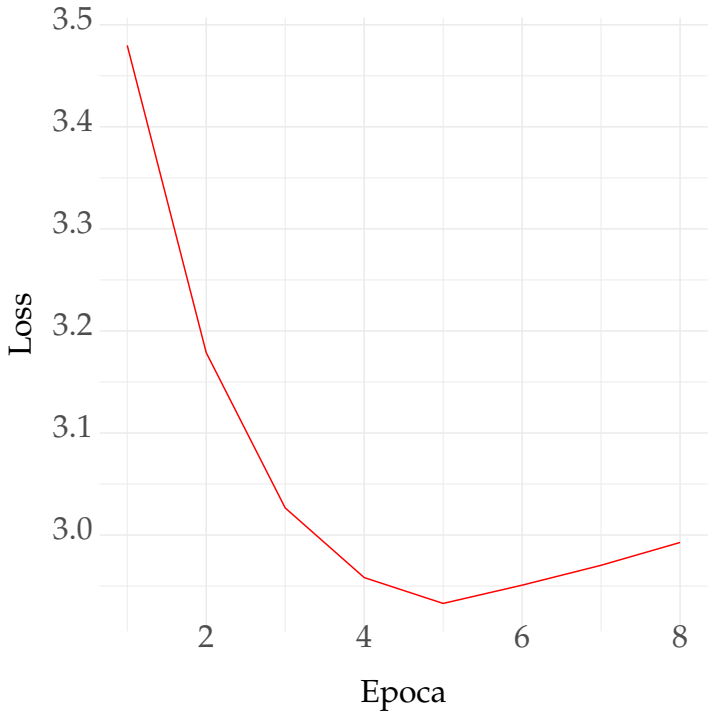

Figura 4.2: Grafico dell'andamento della Loss (T5 Base)

buon livello di convergenza. Tuttavia, nelle epoche successive si nota un leggero incremento, che potrebbe suggerire un inizio di overfitting, con una riduzione della capacità di generalizzazione del modello.

Complessivamente, il modello T5 Base ha mostrato un'efficace ottimizzazione durante le fasi iniziali dell'addestramento. Nelle epoche finali, si evidenziano margini di miglioramento per prevenire l'overfitting e garantire una maggiore stabilità delle prestazioni.

Heatmap delle metriche

La Figura 4.3 mostra la rappresentazione visiva delle metriche di valutazione nel corso delle epoche. La heatmap sintetizza i risultati discussi in precedenza, evidenziando chiaramente il miglioramento progressivo delle metriche e la riduzione della Loss.

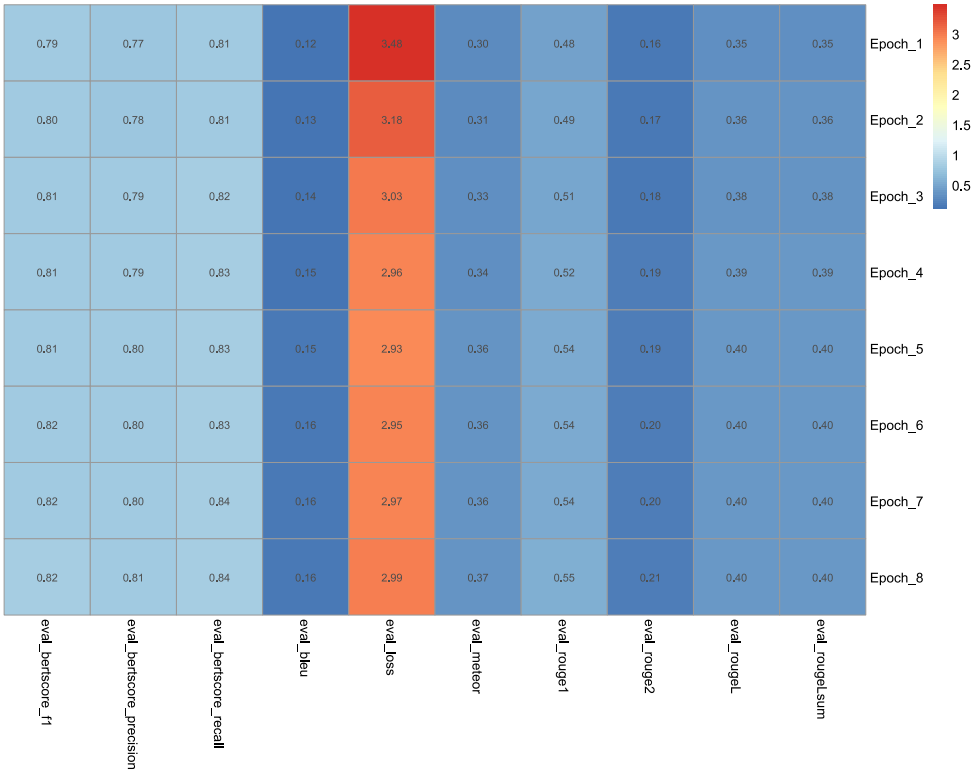

Figura 4.3: Heatmap delle metriche (T5 Base)

4.1.2 Risultati del modello Flan-T5 XXL

Andamento delle metriche

Il grafico di andamento delle metriche (4.4) evidenzia un comportamento simile a quello osservato per il modello T5 Base. Tutte le metriche migliorano progressivamente nelle prime epoche e si stabilizzano nelle epoche finali, segnalando la convergenza del modello.

Le metriche BERTScore mostrano valori elevati e stabili, mentre BLEU e METEOR si attestano su livelli più bassi, riflettendo una minore corrispondenza lessicale precisa. Le metriche ROUGE confermano un solido apprendimento della struttura testuale, con ROUGE-1 e ROUGE-Lsum che raggiungono i valori più alti e ROUGE-2 che rimane più contenuto.

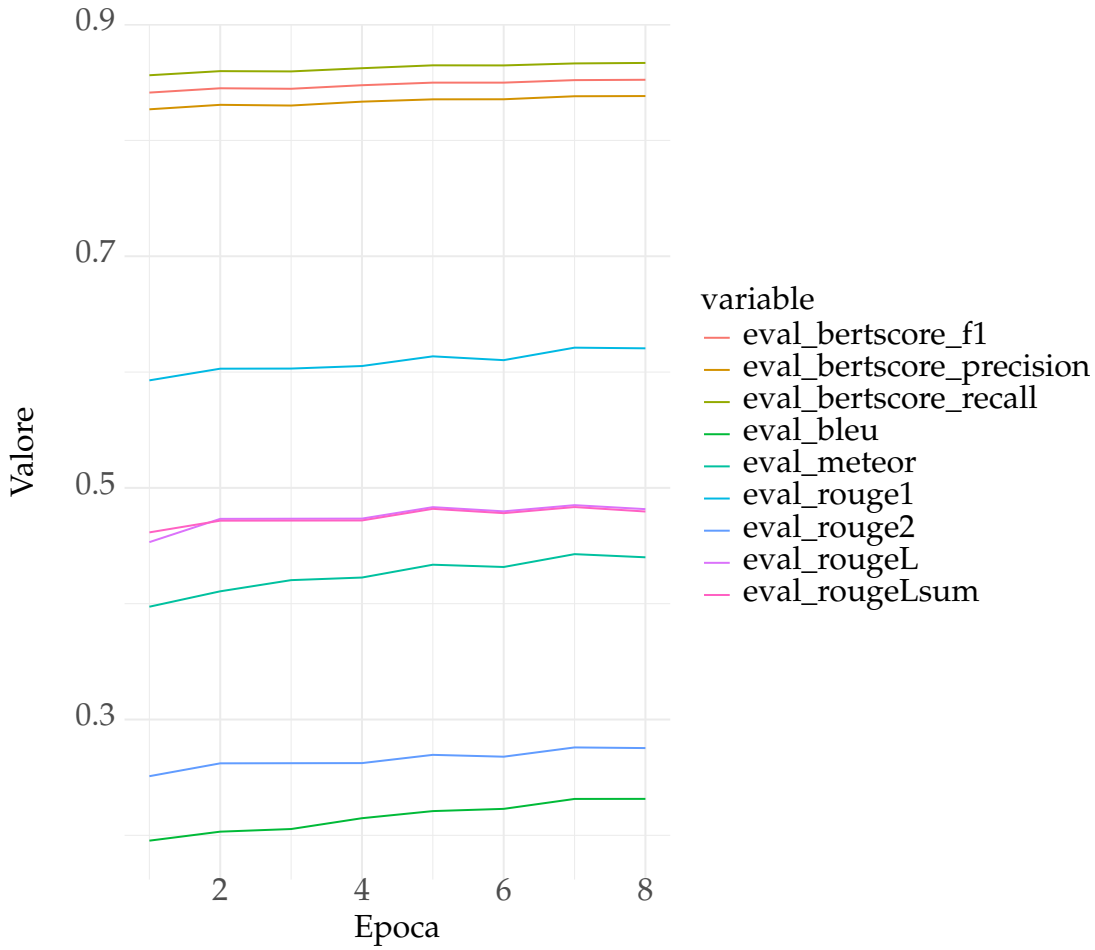

Figura 4.4: Grafico dell'andamento delle metriche (Flan-T5 XXL)

Andamento della Loss

L'andamento della Loss durante l'addestramento del modello Flan-T5 XXL (4.5) mostra una chiara riduzione nelle prime epoche, seguita da un punto minimo intorno alla quarta epoca e da un leggero aumento nelle epoche successive. Questo comportamento è indicativo di un buon apprendimento iniziale, con una possibile tendenza al leggero overfitting nelle fasi finali dell'addestramento.

Rispetto al modello T5 Base, il modello Flan-T5 XXL raggiunge valori di Loss significativamente più bassi, evidenziando una migliore ottimizzazione complessiva. Questo risultato riflette la capacità superiore del modello di apprendere le relazioni tra input e output, probabilmente grazie alla sua maggiore complessità architetturale e ai parametri addizionali disponibili.

Tuttavia, il leggero incremento della Loss nelle epoche finali suggerisce che una regolazione più accurata del processo di addestramento, potrebbe ulteriormente

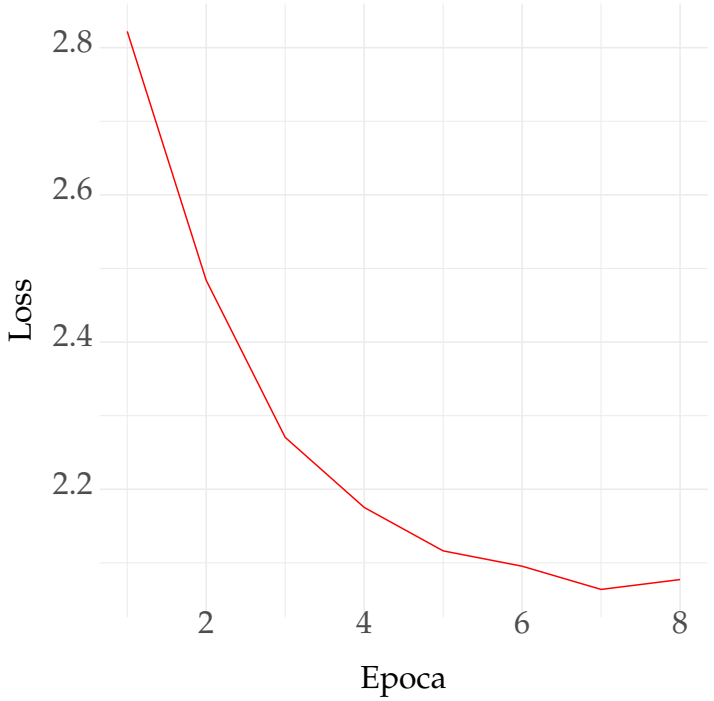

Figura 4.5: Grafico dell'andamento della Loss (Flan-T5 XXL)

migliorare la stabilità del modello e ridurre l'overfitting.

Heatmap delle metriche

La Figura 4.6 fornisce una rappresentazione visiva delle metriche di valutazione per il modello Flan-T5 XXL attraverso le diverse epoche. La heatmap evidenzia un miglioramento costante delle metriche principali, accompagnato da una significativa riduzione della Loss rispetto a T5 Base, sottolineando le migliori capacità di apprendimento del modello.

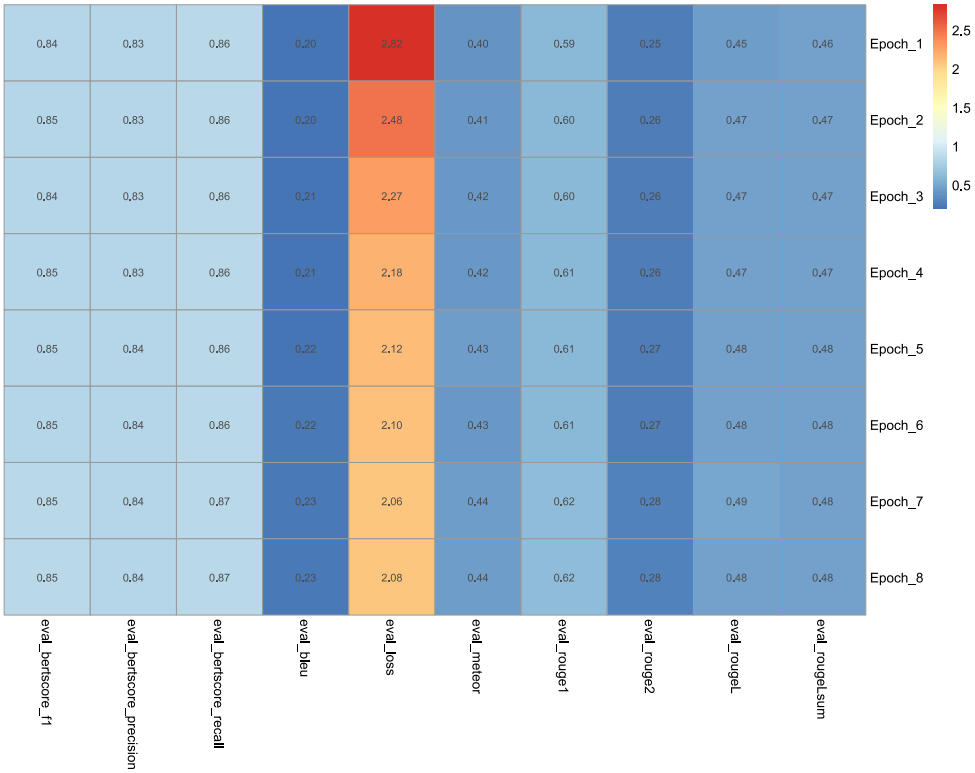

Figura 4.6: Heatmap delle metriche (Flan-T5 XXL)

4.1.3 Risultati del modello LED Large

Andamento delle metriche

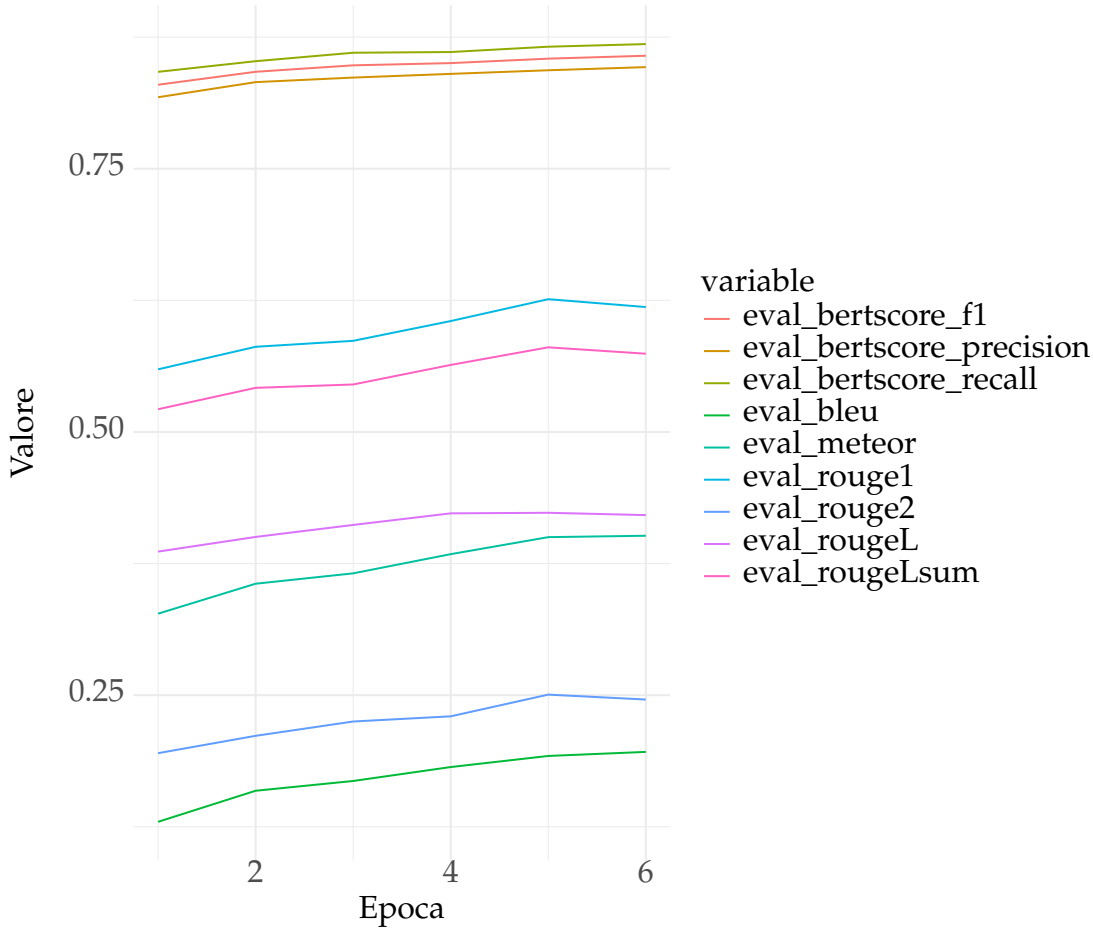

Figura 4.7: Grafico dell'andamento delle metriche (LED Large)

Il grafico di andamento delle metriche (4.7) mette in luce il miglioramento progressivo delle metriche durante le prime epoche di addestramento del modello LED Large, seguito da una fase di stabilizzazione. Rispetto a T5 Base, il modello LED Large si distingue per prestazioni superiori in diverse metriche, evidenziando la sua capacità di gestire input più lunghi in maniera efficace.

Le metriche BERTScore F1, Precision e Recall mostrano un comportamento positivo, raggiungendo valori finali superiori rispetto a T5 Base e simili a quelli ottenuti da Flan-T5 XXL, confermando una buona coerenza semantica tra il testo generato e quello di riferimento. BLEU e METEOR, pur rimanendo su valori inferiori rispetto ad altre metriche, indicano comunque un apprendimento significativo.

Le metriche ROUGE, in particolare ROUGE-1 e ROUGE-Lsum, mostrano risultati comparabili o leggermente superiori rispetto a T5 Base, evidenziando che il modello LED Large è in grado di rappresentare piuttosto efficacemente la struttura globale del testo generato, anche in presenza di input complessi.

Andamento della Loss

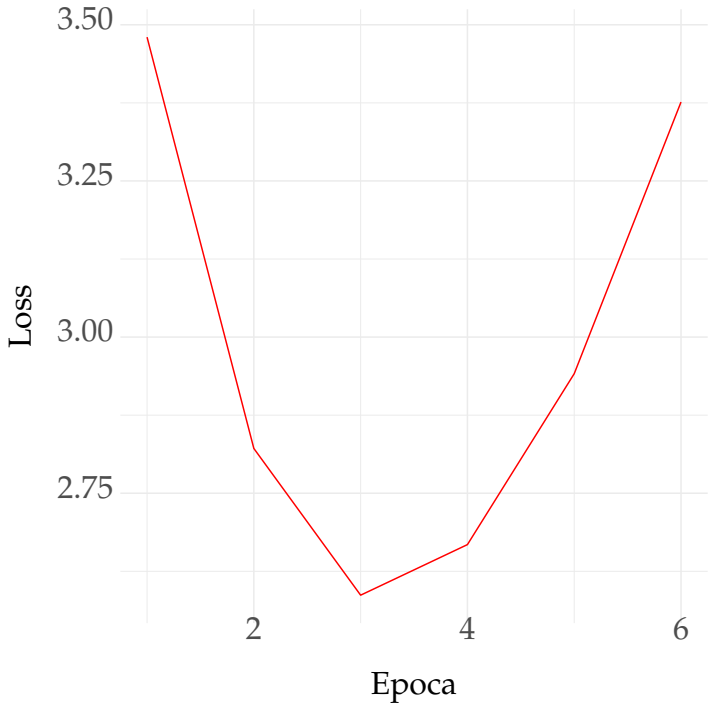

Figura 4.8: Grafico dell'andamento della Loss (LED Large)

Il grafico dell'andamento della Loss (4.8) evidenzia un comportamento interessante durante l'addestramento del modello LED Large. Dopo una fase iniziale di riduzione significativa della Loss, che raggiunge un minimo attorno alla terza epoca, si osserva un aumento nelle epoche successive. Questo comportamento ha portato all'attivazione del criterio di Early Stopping, impostato a 3 epoche senza miglioramenti, fermando l'addestramento alla sesta epoca.

Questo andamento potrebbe essere attribuito a una convergenza instabile, in cui il modello inizialmente si avvicina a un minimo locale, ma successivamente, a causa di oscillazioni nei gradienti o di un Learning Rate elevato, si allontana, aumentando la Loss. Inoltre, la possibilità di overfitting non può essere esclusa: nelle epoche finali,

il modello potrebbe aver iniziato a sovradattarsi ai dati di training, risultando in un peggioramento delle prestazioni sui dati di validazione.

L'attivazione dell'Early Stopping ha probabilmente evitato un peggioramento ulteriore, ma il comportamento osservato indica che potrebbe esserci margine per migliorare la stabilità del training. Nonostante queste instabilità, il LED Large ha dimostrato una buona capacità di apprendimento nelle prime epoche, confermando la sua efficacia per compiti che richiedono l'elaborazione di input lunghi e piuttosto complessi.

Heatmap delle metriche

La Figura 4.9 rappresenta una heatmap delle metriche di valutazione durante l'addestramento del modello LED Large. La visualizzazione evidenzia il miglioramento progressivo delle metriche nelle prime epoche, seguito da una stabilizzazione, con alcune variazioni tra le diverse metriche. La heatmap riflette inoltre l'andamento della Loss, che mostra una riduzione iniziale significativa e un successivo aumento.

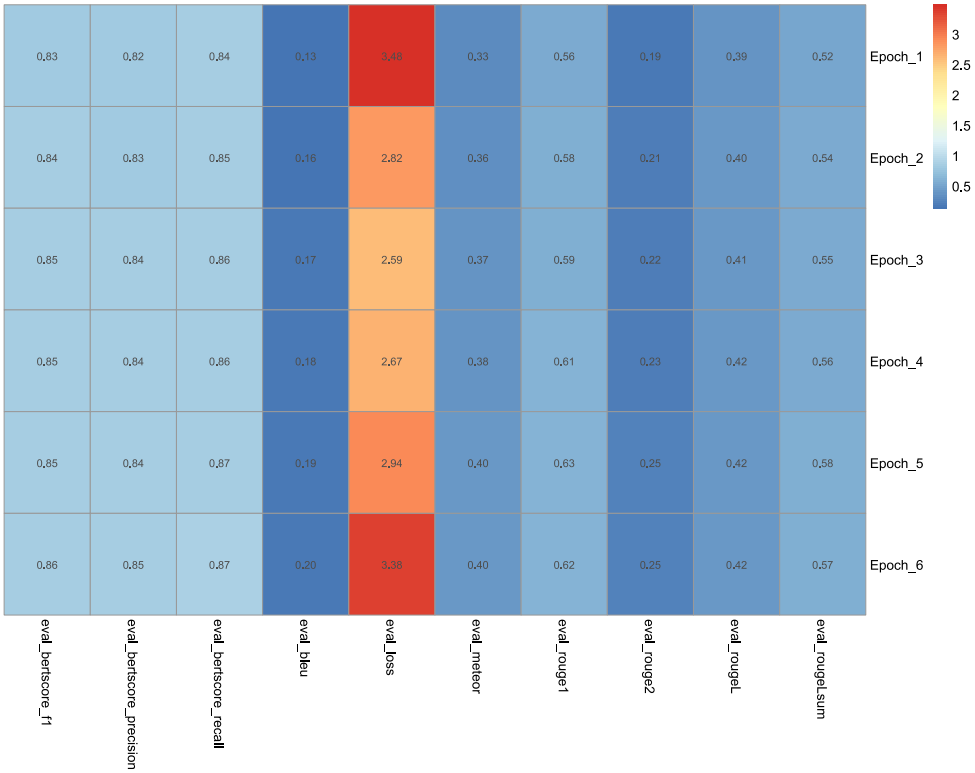

Figura 4.9: Heatmap delle metriche (LED Large)

4.1.4 Confronto dei risultati ottenuti

L'analisi comparativa dei modelli T5 Base, Flan-T5 XXL e LED Large ha evidenziato differenze significative nelle loro prestazioni e comportamenti durante l'addestramento.

Il modello Flan-T5 XXL si è dimostrato il più performante tra i tre, con valori superiori per buona parte delle metriche. Questo risultato conferma la capacità del modello di catturare con maggiore precisione le relazioni semantiche e strutturali tra input e output, probabilmente grazie alla sua architettura avanzata e alla disponibilità di un maggiore numero di parametri. Tuttavia, metriche come BLEU e METEOR rimangono inferiori rispetto ad altre, riflettendo la sfida nella riproduzione esatta delle sequenze di n-grammi.

Il modello LED Large, pur essendo progettato per gestire input molto lunghi,

ha mostrato un comportamento eterogeneo. Anche se meno performante di Flan-T5 XXL in termini di valori assoluti delle metriche, ha superato T5 Base in tutte le valutazioni ad eccezione della Loss. Questo suggerisce che LED Large è stato particolarmente adatto al compito di dover gestire contesti più ampi, superando alcune metriche ottenute con Flan-T5 XXL, pur soffrendo di alcune instabilità durante l'addestramento, come evidenziato dall'andamento della Loss.

Infine, il modello T5 Base si è dimostrato stabile e relativamente efficace, ma con prestazioni complessivamente inferiori rispetto agli altri due. Nonostante l'architettura più semplice, il modello ha comunque mostrato una buona capacità di apprendimento nelle prime epoche, ma con margini di miglioramento più evidenti nelle epoche successive.

Un confronto diretto tra i modelli ha evidenziato che le differenze principali emergono nelle metriche di precisione semantica, come BERTScore, e nelle metriche strutturali, come ROUGE. Flan-T5 XXL si distingue per le sue elevate prestazioni complessive, in particolare nelle metriche semantiche, mentre LED Large mostra un buon equilibrio tra la capacità di catturare contesti estesi e prestazioni competitive, risultando in alcuni casi superiore a Flan-T5 XXL. T5 Base, pur non eccellendo in nessuna metrica specifica, si distingue per la sua semplicità e solidità.

In termini di Loss, Flan-T5 XXL ha raggiunto il livello più basso, riflettendo un'ottimizzazione efficace. LED Large, invece, ha mostrato fluttuazioni che indicano potenziali instabilità, mentre T5 Base ha mantenuto un andamento più prevedibile, ma con valori di Loss più alti.

Nel complesso, questa analisi comparativa ha permesso di delineare i punti di forza e di debolezza di ciascun modello, fornendo indicazioni utili per la selezione del modello più adatto in base ai requisiti specifici del task, come la lunghezza degli input, la precisione semantica richiesta e la complessità computazionale.

4.2 Sviluppi futuri

Di seguito vengono presentate possibili direzioni per migliorare e ampliare il lavoro svolto, con l'obiettivo di affrontare le sfide riscontrate e di identificare nuove opportunità di ricerca.

4.2.1 Utilizzo di modelli più avanzati

Il task di generare revisioni di paper scientifici richiede modelli in grado di catturare la complessità e la ricchezza delle informazioni contenute in documenti strutturalmente e semanticamente complessi e di notevole lunghezza. Modelli come *GPT-4*, *PaLM 3*, *LLaMA 2*, e *Megatron-Turing NLG 530B* rappresentano oggi alcuni tra i modelli più all'avanguardia. Questi modelli, che arrivano ad avere centinaia di miliardi di parametri, grazie alla loro capacità di elaborare input estesi e complessi, rappresentano una soluzione ideale per compiti di elevata complessità, come quello affrontato in questo lavoro.

Il training di questi modelli richiede infrastrutture computazionali estremamente avanzate. Ad esempio, PaLM 540B è stato addestrato utilizzando 3072 TPU v4 chips distribuiti in configurazioni parallele, sfruttando tecniche come il Data e il Model Parallelism [70, 71]. Analogamente, anche il modello Megatron-Turing NLG 530B, addestrato con 2240 GPU NVIDIA A100, ha fatto uso di tecniche di parallelismo avanzato per massimizzare l'efficienza e ridurre i tempi di training. Tali configurazioni consentono di processare dataset enormi e contesti lunghi, garantendo una capacità senza precedenti nella generazione di testi complessi.

Per affrontare le sfide computazionali legate ai modelli di Deep Learning più avanzati, è possibile adottare tecniche di ottimizzazione specifiche. Ad esempio, approcci come l'impiego di LoRA e la Quantizzazione, già implementati con successo in questo lavoro, hanno consentito di ridurre i requisiti di memoria e accelerare l'addestramento, mantenendo comunque un'elevata qualità nel modello Flan-T5 XXL. Inoltre, altri approcci come il Pruning, possono ulteriormente migliorare l'efficienza [72].

L'adozione di modelli più avanzati, insieme all'impiego di tecniche di ottimizzazione, garantirebbe una qualità superiore delle revisioni generate. Purtroppo, il loro utilizzo richiede risorse computazionali che superano di gran lunga quelle disponibili in un contesto accademico triennale.

4.2.2 Utilizzo di dataset e tecniche differenti

La complessità del task di generare revisioni per articoli scientifici deriva anche dalla diversità delle strutture dei paper e delle riviste. Un possibile sviluppo futuro potrebbe prevedere l'uso di dataset più strutturati e specifici, provenienti da riviste scientifiche che adottano formati uniformi. Questo approccio potrebbe semplificare il compito, poiché una migliore strutturazione dei documenti potrebbe mitigare il problema della lunghezza. Tuttavia, la varietà dei formati degli articoli rappresenta una sfida significativa [7]. In tale contesto, l'utilizzo di tecniche di Prompt Engineering potrebbe essere utile per ottimizzare l'interpretazione del contesto da parte del modello, guidando la generazione verso output più mirati e coerenti.

4.3 Conclusioni finali

In conclusione, il complesso task di generare revisioni di paper scientifici è stato affrontato utilizzando diversi modelli di Deep Learning. L'analisi ha evidenziato che Flan-T5 XXL ha ottenuto le migliori prestazioni nelle metriche principali, mentre LED Large si è distinto per la gestione efficace di input lunghi. T5 Base, pur meno performante, ha confermato la sua affidabilità grazie alla semplicità e stabilità, caratteristiche che lo rendono facilmente utilizzabile su buona parte delle configurazioni hardware.

Le tecniche di ottimizzazione hanno permesso di ridurre i requisiti computazionali, dimostrando l'efficacia di approcci praticabili anche in contesti con risorse limitate. Tuttavia, l'adozione di dataset che rispettino una strutturazione più standard e suddivisa, anziché contenere tutto il testo di ciascun paper o revisione in un unico blocco, potrebbe facilitare il miglioramento delle prestazioni, specialmente nell'affrontare la complessità e la lunghezza tipiche dei contenuti scientifici.

Nonostante le risorse disponibili, i risultati ottenuti confermano il potenziale del Deep Learning nel supportare la generazione di contenuti scientifici, aprendo interessanti prospettive per futuri sviluppi.

Bibliografia

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762 (Citato alle pagine iii, 21, 26, 28 e 29)

[2] P. U. K. De Silva and C. K. Vance, *Preserving the Quality of Scientific Research: Peer Review of Research Articles*. Cham: Springer International Publishing, 2017, pp. 73–99. [Online]. Available: https://doi.org/10.1007/978-3-319-50627-2_6 (Citato alle pagine 1 e 6)

[3] V. Warne, "Rewarding reviewers – sense or sensibility? a wiley study explained," *Learned Publishing*, vol. 29, pp. 41–50, 01 2016. (Citato a pagina 1)

[4] M. Kovanis, R. Porcher, P. Ravaud, and L. Trinquart, "The global burden of journal peer review in the biomedical literature: Strong imbalance in the collective enterprise," *PLoS ONE*, vol. 11, 11 2016. (Citato a pagina 1)

[5] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are

few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165 (Citato alle pagine 2, 21, 22, 25 e 27)

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023. [Online]. Available: https://arxiv.org/abs/1910.10683 (Citato alle pagine 2, 22, 25, 33, 39, 41 e 49)

[7] G. D. e. a. Tennant JP, Dugan JM, "A multi-disciplinary perspective on emergent and future innovations in peer review," *F1000Research*, vol. 6, p. 1151, 2017, [version 3; peer review: 2 approved]. [Online]. Available: https://doi.org/10.12688/f1000research.12037.3 (Citato alle pagine 2 e 74)

[8] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020. [Online]. Available: https://arxiv.org/abs/2004.05150 (Citato alle pagine 2, 40 e 41)

[9] S. Ruder, "Neural transfer learning for natural language processing," *PhD Thesis, National University of Ireland, Galway*, 2019. [Online]. Available: https://researchrepository.universityofgalway.ie/server/api/core/bitstreams/db70a7a4-836b-4161-9269-e979efdd01ef/content (Citato alle pagine 2 e 26)

[10] J. Kelly, T. Sadeghieh, and K. Adeli, "Peer review in scientific publications: benefits, critiques, & a survival guide," *EJIFCC*, vol. 25, no. 3, pp. 227–243, 2014. (Citato a pagina 6)

[11] F. Godlee, "Making review articles scientific: the case for the systematic review," *BMJ*, vol. 321, no. 7261, pp. 6–7, 2000. (Citato a pagina 6)

[12] W. Baxt, J. Waeckerle, J. Berlin, and M. Callaham, "Who reviews the reviewers? feasibility of using a fictitious manuscript to evaluate peer reviewer performance," *Annals of Emergency Medicine*, vol. 32, no. 3 Pt 1, pp. 310–317, 1998. (Citato a pagina 7)

[13] Wiley Author Services, "The peer review process," https://authorservices.wiley.com/Reviewers/journal-reviewers/what-is-peer-review/the-peer-review-process.html, accessed: 2024-09-27. (Citato a pagina 8)

[14] S. Cuschieri, "The consort statement," *Saudi Journal of Anaesthesia*, vol. 13, no. Suppl 1, pp. S27–S30, 2019. (Citato a pagina 9)

[15] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher, "The prisma 2020 statement: an updated guideline for reporting systematic reviews," *BMJ*, vol. 372, p. n71, 2021. (Citato a pagina 9)

[16] Wiley Author Services, "How to perform a peer review," https://authorservices.wiley.com/Reviewers/journal-reviewers/how-to-perform-a-peer-review/index.html, accessed: 2024-09-27. (Citato a pagina 10)

[17] ——, "Types of peer review," https://authorservices.wiley.com/Reviewers/journal-reviewers/what-is-peer-review/types-of-peer-review.html, accessed: 2024-09-27. (Citato a pagina 10)

[18] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, p. 160, 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00592-x (Citato a pagina 12)

[19] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Elsevier, 2012. (Citato a pagina 12)

[20] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press, 2012. (Citato a pagina 13)

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, 05 2015. (Citato alle pagine 14, 16 e 17)

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. (Citato a pagina 15)

[23] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008. (Citato alle pagine 21 e 23)

[24] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016. [Online]. Available: https://arxiv.org/abs/1609.08144 (Citato a pagina 21)

[25] B. Liu, "Sentiment analysis and opinion mining," vol. 5, 05 2012. (Citato a pagina 22)

[26] D. Perez-Marin and I. Pascual-Nieto, *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices: Techniques and Effective Practices*, ser. Premier reference source. Information Science Reference, 2011. [Online]. Available: https://books.google.it/books?id=2nUcqtbCOBcC (Citato a pagina 22)

[27] S. A. Abdul-Kader and D. J. Woods, "Survey on chatbot design techniques in speech conversation systems," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 7, 2015. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2015.060712 (Citato a pagina 22)

[28] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, 08 2007. (Citato a pagina 22)

[29] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An easy-to-use framework for state-of-the-art NLP," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, W. Ammar, A. Louis, and N. Mostafazadeh, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019,

pp. 54–59. [Online]. Available: https://aclanthology.org/N19-4010 (Citato a pagina 22)

[30] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703 (Citato alle pagine 22 e 25)

[31] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," 2021. [Online]. Available: https://arxiv.org/abs/2102.07350 (Citato a pagina 23)

[32] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi, "Defending against neural fake news," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/3e9f0fc9b2f89e043bc6233994dfcf76-Paper.pdf (Citato a pagina 23)

[33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781 (Citato a pagina 24)

[34] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162 (Citato a pagina 24)

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805 (Citato alle pagine 24 e 32)

[36] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019. [Online]. Available: https://arxiv.org/abs/1907.11692 (Citato a pagina 24)

[37] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245 (Citato alle pagine 25 e 33)

[38] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*, 3rd ed., 2024, online manuscript released August 20, 2024. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ (Citato a pagina 25)

[39] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," 2022. [Online]. Available: https://arxiv.org/abs/2009.06732 (Citato a pagina 26)

[40] T. Bolukbasi, K.-W. Chang, J. Zou, V. Saligrama, and A. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," 2016. [Online]. Available: https://arxiv.org/abs/1607.06520 (Citato a pagina 26)

[41] T. Sun, A. Gaut, S. Tang, Y. Huang, M. ElSherief, J. Zhao, D. Mirza, E. Belding, K.-W. Chang, and W. Y. Wang, "Mitigating gender bias in natural language processing: Literature review," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1630–1640. [Online]. Available: https://aclanthology.org/P19-1159 (Citato a pagina 26)

[42] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, D. Pedreschi, and F. Giannotti, "A survey of methods for explaining black box models," 2018. [Online]. Available: https://arxiv.org/abs/1802.01933 (Citato a pagina 26)

[43] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," 2019. [Online]. Available: https://arxiv.org/abs/1910.10045 (Citato a pagina 26)

[44] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," 2022. [Online]. Available: https://arxiv.org/abs/2101.03961 (Citato a pagina 27)

[45] G. B. Mohan, R. P. Kumar, P. V. Krishh, A. Keerthinathan, G. Lavanya, M. K. U. Meghana, S. Sulthana, and S. Doss, "An analysis of large language models: Their impact and potential applications," *Knowledge and Information Systems*, 2024. [Online]. Available: https://doi.org/10.1007/s10115-024-02120-8 (Citato a pagina 27)

[46] R. Islam and O. M. Moushi, "Gpt-4o: The cutting-edge advancement in multimodal llm," Jul. 2024. [Online]. Available: http://dx.doi.org/10.36227/techrxiv.171986596.65533294/v1 (Citato a pagina 27)

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997. (Citato a pagina 28)

[48] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014. [Online]. Available: https://arxiv.org/abs/1406.1078 (Citato a pagina 28)

[49] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. (Citato a pagina 28)

[50] D. Rothman, *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, and TensorFlow*. Packt Publishing, 2020. (Citato a pagina 29)

[51] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016. [Online]. Available: https://arxiv.org/abs/1607.06450 (Citato a pagina 29)

[52] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, and W. Pedrycz, "A comprehensive survey on applications of transformers for deep learning tasks," *Expert Systems with Applications*, vol. 241, p. 122666, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423031688 (Citato a pagina 31)

[53] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533 (Citato alle pagine 32 e 37)

[54] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018. [Online]. Available: https://arxiv.org/pdf/1801.06146 (Citato alle pagine 32 e 37)

[55] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685 (Citato a pagina 33)

[56] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017. [Online]. Available: https://arxiv.org/abs/1712.05877 (Citato a pagina 34)

[57] A. S. Luccioni, S. Viguier, and A.-L. Ligozat, "Estimating the carbon footprint of bloom, a 176b parameter language model," 2022. [Online]. Available: https://arxiv.org/abs/2211.02001 (Citato a pagina 34)

[58] Elsevier, "Generative AI Policies for Journals," https://www.elsevier.com/about/policies-and-standards/generative-ai-policies-for-journals, accessed: 2024-11-18. (Citato a pagina 35)

[59] Taylor & Francis, "AI Policy," https://taylorandfrancis.com/our-policies/ai-policy/, accessed: 2024-11-18. (Citato a pagina 36)

[60] V. R. Joseph, "Optimal ratio for data splitting," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 15, no. 4, p. 531–538, Apr. 2022. [Online]. Available: http://dx.doi.org/10.1002/sam.11583 (Citato a pagina 39)

[61] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E. H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," 2022. [Online]. Available: https://arxiv.org/abs/2210.11416 (Citato alle pagine 40 e 41)

[62] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Q. Liu and D. Schlangen, Eds. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6 (Citato a pagina 42)

[63] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703 (Citato a pagina 42)

[64] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101 (Citato a pagina 49)

[65] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," 2018. [Online]. Available: https://arxiv.org/abs/1804.04235 (Citato a pagina 49)

[66] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin, Eds. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 311–318. [Online]. Available: https://aclanthology.org/P02-1040 (Citato a pagina 51)

[67] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013 (Citato a pagina 52)

[68] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, Eds. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909 (Citato a pagina 54)

[69] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020. [Online]. Available: https://arxiv.org/abs/1904.09675 (Citato a pagina 55)

[70] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," 2020. [Online]. Available: https://arxiv.org/abs/1909.08053 (Citato a pagina 73)

[71] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022. [Online]. Available: https://arxiv.org/abs/2204.02311 (Citato a pagina 73)

[72] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2102.00554 (Citato a pagina 73)

Ringraziamenti

Desidero ringraziare il Professor Fabio Palomba per la sua preziosa guida e il Dottor Stefano Lambiase per tutto il supporto fornito durante lo sviluppo di questa tesi.

Un sentito grazie ai miei genitori per il loro sostegno incondizionato e per aver sempre garantito che avessi a disposizione ogni strumento e risorsa necessari per affrontare e proseguire al meglio il mio percorso di studi.

Un ringraziamento speciale va anche ai miei amici, a quelli che hanno vissuto con me l'esperienza universitaria e a quelli che, pur lontani da questo percorso, mi hanno sostenuto con la stessa forza e dedizione. Con il loro supporto, le loro esperienze e le loro personalità uniche e insostituibili hanno contribuito ad arricchire le mie prospettive di vita e a rendere questo percorso più completo e significativo. Non dimenticherò mai le interminabili ore di studio condivise, le notti insonni tra disperazione e determinazione, le lunghe conversazioni che hanno accompagnato momenti di riflessione e crescita, e tutto il tempo trascorso insieme.

Vi ringrazio di cuore, uno ad uno. Grazie.

Questa tesi ha contribuito a piantare un albero di mango in Madagascar tramite il progetto Treedom.

https://www.treedom.net/it/user/antonio-dauria-4038/trees/P3M-JP6J