UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Fine-tuning di un Large Language Model per la definizione di un chatbot capace di operare in un dialetto marchigiano

Fine-tuning a Large Language Model to define a chatbot capable of operating in a Marche dialect

Relatore

Prof. Domenico Ursino

Candidata

Laura Pistagnesi

Correlatori

Dott. Michele Marchetti

Dott. Luca Virgili

ANNO ACCADEMICO 2023-2024

Fate le cose difficili.
Più difficili sono e meglio è.
Quando dite:"Non ce la faccio, questo è troppo difficile".
Quello dovete fare.
Le cose difficili dovete fare.
E se sbagliate non vi preoccupate.
Sbagliate e risbagliate, provate e riprovate.
Gli errori sono necessari, utili e qualche volta anche belli.

Roberto Benigni

Sommario

I Large Language Model rappresentano una rivoluzione nel campo del Natural Language Processing, offrendo soluzioni avanzate che consentono alle macchine di comprendere, generare e interagire con il linguaggio umano. Tuttavia, l'adattamento di questi modelli a lingue e dialetti caratterizzati da scarsità di risorse rappresenta una sfida ancora aperta. Questa tesi si inserisce in questo panorama esplorando il caso del dialetto fermano, utilizzato come esempio rappresentativo per valutare le potenzialità degli LLM in contesti linguistici poco rappresentati. Grazie alla costruzione di un dataset dialettale specifico e allo sviluppo di pipeline per il fine-tuning e la valutazione, la ricerca si è focalizzata su tre task distinti: generazione di testo, Target Word Prediction e Target Word Selection. I risultati ottenuti offrono uno spunto critico sui limiti attuali degli LLM nel comprendere e adattarsi a contesti dialettali, proponendo al contempo strategie di ottimizzazione e possibili applicazioni pratiche per la preservazione e valorizzazione di lingue e dialetti sottorappresentati.

Keyword: Natural Language Processing, Large Language Model, Dialetti, Fine-tuning, BERT, Minerva, Chatbot

Indice

Elenco delle figure

Elenco delle tabelle

Listings

Introduzione

Gli ultimi decenni hanno segnato una rivoluzione nel campo del Natural Language Processing (NLP), una disciplina che ha l'obiettivo di sviluppare tecnologie che consentano alle macchine di comprendere, generare e interagire con il linguaggio umano. Strumenti come i motori di ricerca, i traduttori automatici e gli assistenti vocali hanno rivoluzionato settori cruciali come l'educazione, la sanità e il commercio, migliorando significativamente l'accesso alle informazioni e l'efficienza dei servizi.

In questo contesto, i Large Language Model (LLM) rappresentano una delle innovazioni più significative, offrendo strumenti capaci di apprendere in modo autonomo relazioni semantiche complesse, analizzare enormi volumi di dati testuali e generare contenuti con una fluidità e coerenza senza precedenti. A differenza dei sistemi tradizionali, basati su regole fisse o intent detection, gli LLM comprendono il contesto in modo più profondo, offrono risposte articolate e altamente contestualizzate, gestiscono conversazioni dinamiche e si adattano in tempo reale a richieste personalizzate, rivoluzionando applicazioni come assistenti vocali, chatbot avanzati e strumenti di sintesi e generazione testuale.

Nonostante queste potenzialità, gli LLM si trovano ad affrontare un problema fondamentale, ovvero la scarsità di risorse linguistiche per alcune lingue e contesti specifici. Mentre le lingue principali, come l'inglese o il cinese, beneficiano di dataset vasti e diversificati, le lingue regionali, le minoranze linguistiche e i dialetti sono spesso esclusi dal panorama dell'innovazione tecnologica. Questo non solo limita la loro rappresentazione, ma perpetua un divario tecnologico che penalizza le comunità meno rappresentate. Il lavoro di adattamento degli LLM a contesti linguistici caratterizzati da scarsità di risorse non è solo una sfida tecnica, ma un'opportunità per democratizzare l'accesso alle tecnologie avanzate e preservare la diversità culturale.

Le motivazioni alla base di questa tesi risiedono nella volontà di affrontare queste problematiche, esplorando come i modelli pre-addestrati possano essere adattati a contesti con risorse limitate. L'obiettivo non è solo quello di migliorare la capacità dei modelli di comprendere e generare testi, ma anche di contribuire allo sviluppo di soluzioni che abbiano un impatto sociale e culturale positivo.

La presente tesi si colloca in questo ambito di ricerca, focalizzandosi sull'adattamento di alcuni LLM ad un dialetto marchigiano, in particolare il fermano, come caso di studio rappresentativo.

La valorizzazione e la preservazione del dialetto fermano rappresentano un ulteriore pilastro di questa ricerca. In un'epoca in cui le lingue minori rischiano di scomparire, adattare le tecnologie più avanzate a tali contesti offre un'opportunità concreta per conservarne l'uso e la memoria.

L'obiettivo non è solo quello di sviluppare strumenti tecnici destinati, ad esempio, al turismo, ma anche di creare una connessione tra innovazione tecnologica e tutela del patrimonio culturale, dimostrando come l'Intelligenza Artificiale possa essere un alleato nella sostenibilità linguistica.

La presente tesi si colloca, quindi, in un ambito di ricerca innovativo, proponendo un caso di studio rappresentativo per esplorare come i modelli linguistici avanzati possano essere adattati a contesti sottorappresentati. Questo esperimento non mira solo a migliorare le capacità dei modelli nella comprensione e generazione di testo, ma anche a sviluppare soluzioni con un impatto sociale e culturale.

La tesi si articola in diverse fasi, ognuna delle quali è pensata per illustrare in modo dettagliato e sistematico le attività svolte e gli obiettivi perseguiti nel corso del lavoro.

La prima fase si concentra sull'analisi teorica degli LLM con un approfondimento sui loro principi di funzionamento e sulle principali architetture che li compongono, come i Transformer. Questo step consente di comprendere le potenzialità e i limiti di tali modelli, gettando le basi per il loro utilizzo in contesti dialettali. Si analizzano, inoltre, le applicazioni e i vantaggi derivanti dall'adozione di queste tecnologie, evidenziando come il loro impiego possa estendersi anche a lingue meno rappresentate.

Successivamente, vengono definiti i requisiti del progetto, identificando tutti gli elementi essenziali per sviluppare una soluzione capace di comprendere e rispondere in un dialetto. Questo processo mira a stabilire obiettivi concreti e realistici, basandosi sulle risorse disponibili e sulle specifiche esigenze del contesto applicativo.

La fase di implementazione si articola in una serie di attività mirate. Si parte dalla raccolta dei dati, un'operazione complessa che include l'estrazione, la pulizia e il pre-processing di risorse dialettali. A seguire, vengono definiti tre task principali per la valutazione delle capacità dei modelli: generazione di testo, Target Word Prediction (TWP) e Target Word Selection (TWS). Ogni task richiede approcci specifici, dall'adattamento dei dati alla personalizzazione di pipeline di pre-processing e metriche di valutazione. In questa fase, sono stati individuati e selezionati due modelli risultati i più conformi ai requisiti: Minerva e BERT. Entrambi i modelli sono stati sottoposti a un processo di fine-tuning utilizzando il dataset dialettale creato appositamente.

Infine, il lavoro prevede un'analisi approfondita dei risultati, sia qualitativi che quantitativi, per valutare le performance dei modelli e individuare aree di miglioramento. Inoltre, vengono proposte prospettive di ottimizzazione e suggerimenti per applicazioni future, con l'obiettivo di ampliare il campo d'applicazione dei modelli a ulteriori contesti linguistici scarsamente rappresentati.

La presente tesi è composta da cinque capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 vengono introdotti i Large Language Model, con un approfondimento sulle loro architetture, sul loro funzionamento e sulle loro applicazioni, evidenziandone i vantaggi e le sfide.
- Nel Capitolo 2 vengono analizzati i requisiti del sistema, descrivendo le risorse necessarie e conducendo uno studio di fattibilità.
- Nel Capitolo 3 viene descritto in modo dettagliato il processo di implementazione, dalla raccolta dei dati al fine-tuning dei modelli per task specifici.
- Nel Capitolo 4 vengono presentati i risultati dei test, confrontando le performance di Minerva e BERT sui task di Target Word Prediction e Target Word Selection.
- Nel Capitolo 5 vengono discussi i risultati ottenuti, identificando le problematiche riscontrate e proponendo prospettive di ottimizzazione.

CAPITOLO 1

I Large Language Model

In questo capitolo si effettuerà una panoramica sui Large Language Model. Partendo da un'introduzione sulla disciplina del Natural Language Processing, si analizzerà l'evoluzione nel tempo dei Language Model fino ai più recenti Large Language Model. Di questi ultimi verrà, quindi, descritto il funzionamento generale con una particolare attenzione sull'architettura dei Transformer. Verrà, poi, proposta una tassonomia in base a diversi fattori come architettura e task. Infine, verrano presentati le applicazioni, i vantaggi e le sfide dei Large Language Model.

1.1 Introduzione ai Large Language Model

Gli esseri umani sono in grado di comunicare tra loro, di esprimere opinioni, emozioni e pensieri attraverso il linguaggio naturale. Con l'avanzare degli strumenti e delle tecnologie a disposizione, si cerca di affrontare sfide sempre più impegnative. Una di queste è la capacità delle macchine di comprendere ed interagire con il linguaggio umano.

Il Natural Language Processing (NLP) è un'area dell'Intelligenza Artificale che si occupa di progettare e implementare metodi e algoritmi per analizzare, modellare e comprendere il linguaggio umano.

L'importanza di questa disciplina è giustificata dalla vasta gamma di applicazioni in molteplici contesti. Alcuni esempi di task di cui si occupa il Natural Language Processing sono:

- la modellazione del linguaggio, il cui scopo è prevedere la parola successiva in una frase in base al contesto;
- la classificazione del testo in categorie;
- l'estrazione di informazioni rilevanti da un determinato testo;
- il recupero di informazioni e documenti pertinenti ad una determinata query;
- lo sviluppo di agenti conversazionali in grado di interagire attraverso il linguaggio naturale;
- il riassunto dei testi.

Gli strumenti principali per affrontare i task di Natural Language Processing sono i Language Model (LM). Gli LM sono modelli computazionali in grado di comprendere e generare il linguaggio umano imparando la probabilità di sequenze di parole. Questi modelli analizzano testi di riferimento per apprendere i pattern, le strutture e le relazioni tra le parole o frasi, e prevedere quale parola o frase potrebbe seguire un'altra.

Grazie all'aumento della potenza computazionale e alla disponibilità di vasti dataset, i Language Model si sono evoluti in modelli molto più complessi e potenti, noti come Large Language Models (LLM). Il termine *Large* si riferisce al numero di parametri nel modello (nell'ordine dei miliardi) e alla quantità dei dati di training. Grazie a queste caratteristiche, tali modelli sono in grado di comprendere più profondamente sfumature linguistiche, contesto e semantica delle frasi e dimostrano una buona generalizzazione in un'ampia gamma di attività.

1.1.1 Evoluzione dei Large Language Model

La Figura 1.1 fornisce una panoramica dell'evoluzione dei Large Language Model.

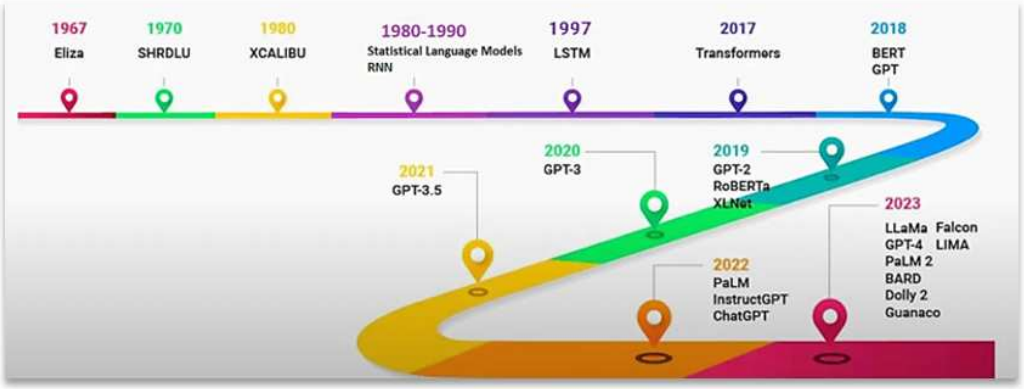


Figura 1.1: Breve rappresentazione dell'evoluzione degli LLM

Uno dei primi tentativi di interazione uomo-macchina mediante linguaggio naturale risale agli anni '60, quando Joseph Weizenbaum sviluppò ELIZA, un "chatbot" progettato per simulare una conversazione con uno psicoterapeuta, con risposte semplici, basate su regole. Il sistema rispondeva riflettendo e riformulando le affermazioni ricevute, creando un'illusione di dialogo empatico. Ad esempio, se un utente digitava: "Sono triste perché nessuno mi capisce", un possibile output di ELIZA era "Perché pensi che nessuno ti capisca?". Sebbene privo di una vera comprensione del contesto, questo esperimento segnò un punto di partenza fondamentale per tutti i successivi studi.

Successivamente, l'NLP è evoluto grazie alla possibilità di sfruttare i modelli matematici statistici per affrontare le prime sfide di analisi del linguaggio naturale e identificazione di pattern linguistici.

Alcuni esempi di Statistical Language Model utilizzati nell'NLP sono i seguenti:

- *Modelli a n-grammi*, prevedono una parola in base alle probabilità condizionate delle *n* parole precedenti (un-bigram considera la parola precedente, mentre un trigramme le ultime due). I modelli a n-grammi sono stati tra i primi tentativi di modellazione linguistica, ma soffrono di limiti di contesto, poiché le probabilità dipendono solo da una finestra limitata di parole.

- *Naive Bayes*, è un algoritmo di classificazione supervisionata che individua la probabilità di osservare un'etichetta di classe dato l'insieme delle feature dei dati di input. Questo

algoritmo calcola la distribuzione di probabilità condizionale $P(\text{label}|\text{text})$ utilizzando il teorema di Bayes:

$$P(\text{label}|\text{text}) = \frac{P(\text{label}) \cdot P(\text{text}|\text{label})}{P(\text{text})}$$

L'algoritmo predice l'etichetta con la massima probabilità congiunta, assumendo che le parole nel testo siano indipendenti, quindi:

$$P(\text{text}|\text{label}) = P(\text{word}_1|\text{label}) \cdot P(\text{word}_2|\text{label}) \cdot \cdots \cdot P(\text{word}_n|\text{label})$$

Nell'NLP, questa tecnica è utile per problemi come il rilevamento di spam o la ricerca di errori in un codice software. Nonostante l'ipotesi forte di indipendenza, Naive Bayes è comunemente utilizzato come algoritmo di partenza per la classificazione dei testi.

- *Hidden Markov Models (HMM)*, sono modelli probabilistici che introducono degli stati nascosti. In NLP, gli HMM sono spesso applicati per attività come il Part-of-Speech (POS) tagging, dove ogni parola di una frase rappresenta uno stato osservabile e l'etichetta grammaticale (il POS) è lo stato nascosto. Gli HMM utilizzano due componenti principali, ovvero la probabilità di transizione, cioè la probabilità di passare da uno stato nascosto a un altro (ad esempio, da un verbo a un sostantivo) e la probabilità di emissione cioè la probabilità che uno stato nascosto generi una particolare osservazione (come la probabilità di vedere una certa parola per un dato POS). Gli HMM sono importanti in quanto il linguaggio naturale è intrinsecamente sequenziale.

Con l'evolversi del machine learning, anche le principali tecniche di apprendimento, come Support Vector Machine e Decision Tree, sono state sfruttate nel settore dell'NLP.

Nonostante l'efficacia di questi approcci, i loro limiti sono diventati evidenti per compiti più complessi, come comprendere il significato più profondo in frasi lunghe o contestualizzare il linguaggio su larga scala.

Nella fine del XX secolo, l'ascesa delle reti neurali ha portato ad un grande passo in avanti nell'evoluzione del Natural Language Processing. Tra queste, le reti neurali ricorrenti (RNN) sono emerse per prime nel 1986 e si sono rapidamente diffuse in tutto il mondo. A differenza delle reti neurali feedforward convenzionali, che consentono il flusso di informazioni in una sola direzione, nelle RNN gli output di alcuni layer vengono dati in input agli strati precedenti. In questo modo, viene introdotto il concetto di memoria, cioè le unità neurali sono in grado di ricordare ciò che hanno elaborato fino a quel momento e le informazioni vengono aggiornate a ogni passo temporale. Le RNN, grazie a questo meccanismo di apprendimento sequenziale, funzionano molto bene per risolvere una serie di task di NLP come la classificazione di testi e la traduzione automatica. Lo svantaggio delle RNN è il problema della "memoria smemorata", cioè non sono in grado di ricordare lunghi contesti, quindi non funzionano bene con frasi lunghe.

Nel 1997 sono state sviluppate le Long Short-Term Memory (LSTM) per mitigare questo difetto delle RNN. Tali reti sono costituite da un layer che determina le informazioni da memorizzare, scartare o inviare in uscita ad ogni passo. In questo modo il contesto irrilevante non viene considerato e viene ricordata solo la parte del testo necessaria per il task che si sta effettuando, alleggerendo il carico da memorizzare.

Queste architetture hanno rappresentato innovazioni significative, ma nel 2017, con il paper "Attention Is All You Need" (Vaswani, Shazeer, Parmar, et al.), i Transformer hanno rivoluzionato il campo dell'NLP introducendo il meccanismo di self-attention multi-head. Piuttosto che modellare il contesto in modo sequenziale, i Trasformer analizzano contemporaneamente più parti del contesto valutando la relazione tra ogni elemento o parola di una sequenza di input e le altre parole della sequenza stessa.

I Transformer hanno reso possibile l'elaborazione parallela di sequenze e, di conseguenza, l'emergere dei Large Language Model (LLM).

Dal 2018 in poi, questi modelli hanno raggiunto una portata ancora maggiore grazie alla disponibilità di dataset enormi e risorse computazionali avanzate, permettendo loro di svolgere attività complesse con precisione crescente e di dimostrare capacità avanzate nella comprensione e generazione del linguaggio naturale.

1.2 Principi di funzionamento

I Large Language Model si basano su principi avanzati di deep learning, sfruttando architetture di reti neurali per analizzare e comprendere il linguaggio umano. Questi modelli sono addestrati su dataset di enormi dimensioni utilizzando il self-supervised learning, cioè tecniche di auto-apprendimento di schemi e relazioni all'interno dei dati. Il testo in input pre-processato attraversa i vari livelli della rete. In ogni strato, il modello aggiunge e rifinisce la comprensione del contesto, contribuendo alla generazione di predizioni accurate. Durante l'addestramento, gli LLM acquisiscono la capacità di identificare e memorizzare dipendenze complesse (relazioni semantiche e sintattiche) tra parole, frasi e contesti.

Gli LLM sono costituiti da molteplici livelli e componenti che variano in base al tipo di architettura.

1.2.1 Transformer

La maggior parte dei Large Language Model si basano sull'architettura dei Transformer.

I Transformer sono una tipologia di rete neurale introdotta nel 2017 da Vaswani et al. nel famoso articolo *"Attention is All You Need"*. La rivoluzione introdotta in tale articolo è il meccanismo della self-attention. Questo permette di valutare ogni token di una sequenza in input in relazione a tutti gli altri token della sequenza. Osservando tutte le parole circostanti, la self-attention permette di comprendere il contesto completo delle parole risolvendo il problema dell'ambiguità di linguaggio.

I modelli precedenti ai Transformer processavano le parole in sequenza. La sequenzialità comporta che le parole non ancora osservate all'interno del contesto, non vengono prese in considerazione quando, in realtà, potrebbero cambiare il significato di altre parole. I Transformer, invece, sono in grado di elaborare tutte le parole in un testo simultaneamente tenendo conto di tutti gli elementi della sequenza. Questo migliora anche le prestazioni computazionali e rende l'architettura dei Transformer particolarmente adatta a lavorare su dataset di grandi dimensioni, accelerando notevolmente l'addestramento.

L'architettura dei Transformer è mostrata in Figura 1.2 e si compone di due parti: un encoder e un decoder. L'encoder mappa una sequenza di rappresentazioni simboliche in input $(x_1, \ldots, x_n)$ in una sequenza di rappresentazioni continue $z = (z_1, \ldots, z_n)$. A partire da z, il decoder genera una sequenza di simboli in output $(y_1, \ldots, y_m)$, un elemento alla volta. Ad ogni passaggio, il modello è auto-regressivo, cioè utilizza i simboli generati in precedenza come input aggiuntivo per produrre il successivo.

Ciascuna parte include i seguenti componenti principali:

- *Embedding Posizionale*: poichè i Transformer non processano le parole in sequenza, non si ha alcuna informazione relativa alla posizione occupata dai token nella sequenza iniziale. Questa fase aggiunge tale informazione sotto forma di codifiche posizionali (positional encoding).
- *Multi-Head Attention*: sono dei meccanismi di self-attention che operano in parallelo. Questo approccio permette al modello di catturare molteplici relazioni e contesti tra

le parole in una frase. Nel caso del decoder, si parla di masked multi-head attention poichè si impedisce al modello di accedere alle parole future durante il processo di generazione del testo.

- *Feed-Forward Layer*: ciascun layer dell'encoder e del decoder contiene una rete completamente connessa che consente di rafforzare le relazioni non lineari tra i dati.
- *Strato di normalizzazione e somma*: dopo ogni livello di attenzione e ogni livello feed-forward, si aggiunge il valore originale dell'input e si applica una normalizzazione per stabilizzare il processo di addestramento. Questo passaggio migliora il flusso dei gradienti (evitando che il segnale diminuisca durante la backpropagation) e facilita l'addestramento riducendo il rischio di overfitting.
- *Softmax e Layer Linear per l'output*: alla fine del decoder, c'è uno strato lineare seguito da uno strato softmax, che converte le rappresentazioni finali in probabilità. Questo passaggio consente al modello di generare il token successivo con la massima probabilità.

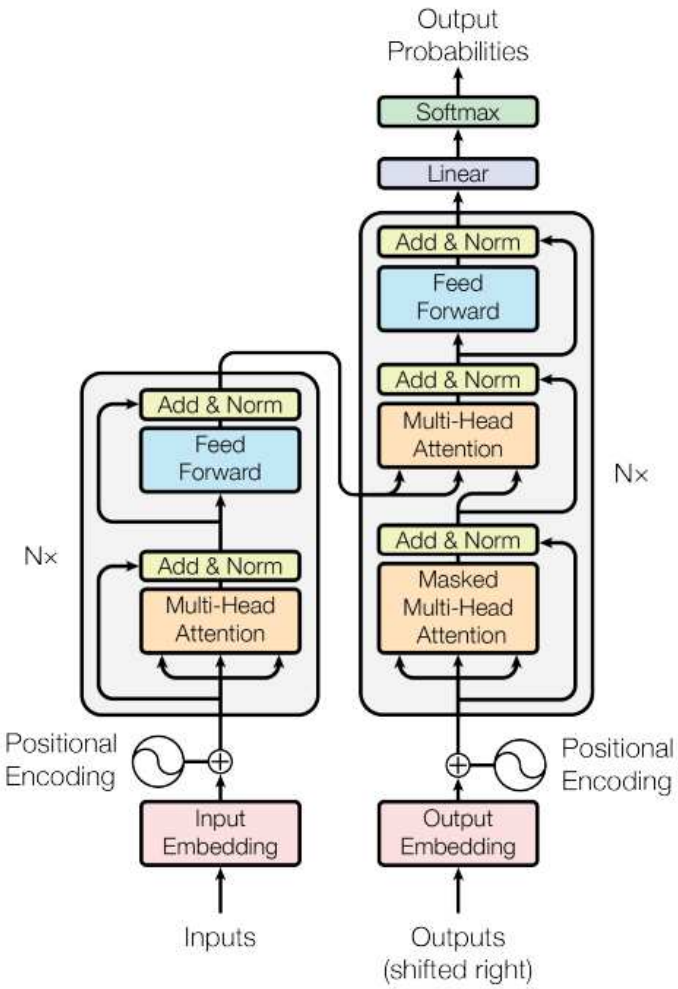


Figura 1.2: Architettura dei Transformer

1.2.2 Architetture principali

I Large Language Model, in base alla loro architettura, possono essere suddivisi in diverse categorie:

- *Encoder-only*: sono progettati per codificare l'input e creare rappresentazioni di alta qualità che catturano le caratteristiche rilevanti del testo. Questi modelli sono particolarmente efficaci per i compiti di classificazione, estrazione di entità e ricerca semantica, dove l'obiettivo principale è rappresentare il contesto in modo accurato senza generare testo in sequenza. Un esempio di modello che adotta questa architettura è BERT.
- *Decoder-only*: sono progettati per generare testo in modo autoregressivo, cioè parola per parola, utilizzando solo le informazioni già generate fino a quel momento. Questa architettura è ideale per i compiti di generazione di testo, completamento e risposte a domande aperte, in cui il modello deve produrre un output continuo basato su un contesto iniziale. I modelli GPT sono decoder-only.
- *Encoder-Decoder*: combinano le caratteristiche di entrambe le architetture, con un encoder che elabora l'input per generare una rappresentazione e un decoder che genera l'output basandosi su tale rappresentazione. Questa struttura è adatta ai compiti di traduzione automatica, riassunto e altre attività di sequence-to-sequence, dove è necessario sia comprendere il testo di partenza sia generare un testo in un formato differente. BART è un modello encoder-decoder.

A seconda dell'architettura utilizzata, del tipo di training e dei task specifici per cui sono progettati, i Large Language Model possono essere:

- *Autoregressive models*: questi modelli generano testo in modo sequenziale prevedendo ogni parola in base agli elementi visti in precedenza. Sono tipicamente modelli decoder-only, ideali per task di generazione del testo.
- *Autoencoding models*: alcune parole sono mascherate all'interno della sequenza in input e il modello viene addestrato a predire le parole mascherate sulla base del contesto circostante. Questo approccio è particolarmente efficace per i compiti di comprensione e rappresentazione del testo, come la classificazione e l'analisi del sentiment, dove è cruciale catturare l'intero contesto senza necessariamente generare nuovo testo.
- *Sequence-to-sequence models*: sono tipicamente basati su architetture encoder-decoder e vengono utilizzati per trasformazioni complesse da un input sequenziale a un output sequenziale di diverso tipo. Sono ottimizzati, quindi, per task di traduzione automatica e per la sintesi di testi.
- *Multimodal models*: sono progettati per gestire in input dati di diverso tipo come testi, immagini, video, ecc. Possono essere utilizzati per task complessi come la descrizione delle immagini o la generazione di immagini basate su input testuali.
- *Retrieval-based models*: combinano la generazione di testo con la capacità di recuperare informazioni da un database esterno o da un corpus di documenti. Sono modelli avanzati per la ricerca di informazioni in grado di rispondere a domande specifiche in modo ottimale.

1.3 Applicazioni

I Large Language Model stanno dimostrando la loro importanza in molti ambiti e applicazioni.

Nel settore delle vendite, gli LLM possono essere utilizzati per la creazione di chatbot per l'assistenza clienti con funzionalità avanzata, ottime capacità di comprensione, generazione e padronanza del linguaggio naturale.

L'uso di modelli di tipo encoder-decoder ha portato notevoli miglioramenti nella traduzione automatica.

In ambito medico, i Large Language Model possono essere impiegati per analizzare e sintetizzare informazioni provenienti da una vasta gamma di dati. Questi modelli sono utili sia nello sviluppo di nuovi farmaci, grazie alla capacità di elaborare rapidamente conoscenze scientifiche complesse, sia come supporto per i medici nella formulazione delle diagnosi.

Nel campo giuridico, gli LLM trovano applicazione nell'analisi di documenti legali per l'identificazione e la classificazione di informazioni, come clausule specifiche nei contratti. Un altro esempio è l'uso degli LLM per il legal research, dove il modello ha il compito di effetturare ricerche e recuperare informazioni da leggi e casi simili per supportare avvocati e giudici nella formulazione di argomentazioni e verdetti.

Il fine-tuning rappresenta un'applicazione chiave dei Large Language Model, rendendo questi modelli adattabili a specifici contesti e domini. In fase di addestramento, gli LLM vengono addestrati su ampie raccolte di dati generali per sviluppare una comprensione di base del linguaggio e dei pattern testuali. Tuttavia, per specializzarli in compiti specifici, come la generazione di risposte accurate in un preciso contesto aziendale, il modello può essere affinato tramite fine-tuning su dataset specifici. Durante il fine-tuning, il modello viene ulteriormente addestrato su set di dati che riflettono il linguaggio e i contenuti del dominio target. In questo modo esso sarà specializzato in un campo continuando ad avere le conoscenze generali del pre training. Gli LLM devono, quindi, essere addestrati su enormi dataset una sola volta e poi possono essere riutilizzati per diversi contesti specifici implicando meno risorse computazionali e meno dati.

Queste sono solo alcune delle applicazioni principali degli LLM. La disciplina dell'NLP, in generale, si sta evolvendo sempre più velocemente emergendo in ogni campo della vita quotidiana fornendo strumenti innovativi e vantaggiosi.

L'uso dei Large Language Model richiede sempre un controllo umano qualitativo per verificare la precisione delle risposte generate, soprattutto in settori delicati come quello medico e legale, dove eventuali errori possono portare a gravi conseguenze. Il ruolo di questi modelli non è sostituire le competenze specialistiche, ma piuttosto velocizzare l'esecuzione di attività ripetitive, ridurre i tempi di lavoro e facilitare le attività degli esperti del settore.

1.4 Vantaggi e sfide

I Large Language Model offrono vantaggi significativi; infatti:

- tali modelli hanno un'ottima capacità di comprensione e generazione del linguaggio naturale;
- grazie al fine-tuning, essi possono essere utilizzati e personalizzati facilmente;
- tutti i task lunghi e ripetitivi, come la ricerca di informazioni, la sintesi di testi o l'assistenza nei processi decisionali, possono essere eseguiti da tali modelli riducendo i costi operativi e alleggerendo il lavoro delle risorse umane;
- gli LLM facilitano l'accesso a grandi quantità di informazioni in modo rapido e sintetico;
- gli LLM consentono alle aziende di analizzare in tempi rapidi vasti volumi di dati testuali, come quelli provenienti dai social media, dalle recensioni online e dagli articoli di settore, supportando decisioni strategiche più efficienti e mirate.

Sebbene i vantaggi siano numerosi, gli LLM presentano ancora le sfide classiche di NLP nonché alcuni limiti da considerare; infatti:

- l'ambiguità di linguaggio, la conoscenza comune che contraddistingue gli esseri umani, la creatività e la diversità tra i linguaggi sono i motivi per cui l'NLP è ancora una disciplina sfidante;
- gli LLM richiedono enormi risorse computazionali sia per l'addestramento che per l'inferenza, il che comporta costi significativi in termini di energia e infrastruttura;
- i modelli sono addestrati su grandi quantità di dati provenienti da molteplici fonti che possono contenere bias culturali o sociali;
- possono presentarsi problemi di privacy e sicurezza in quanto i modelli possono memorizzare informazioni sensibili se addestrati su dati contenenti informazioni personali.

CAPITOLO 2

Analisi dei requisiti

Il presente capitolo si concentrerà sull'analisi dei requisiti necessari per la progettazione e lo sviluppo di una soluzione in grado di comprendere e rispondere in un dialetto. Verranno delineati gli elementi essenziali del progetto, suddividendo i requisiti in categorie funzionali e non funzionali, al fine di garantire che il sistema soddisfi le esigenze operative e qualitative attese. Verranno, altresì, definiti gli strumenti e le risorse principali indispensabili per l'implementazione del progetto. Il capitolo si concluderà con uno studio di fattibilità, che valuta i limiti e i vincoli tecnici del sistema.

2.1 Panoramica generale

Lo sviluppo recente dei Large Language Model ha rivoluzionato il campo del Natural Language Processing, offrendo capacità senza precedenti per comprendere, generare, tradurre e modellare testi in una vasta gamma di contesti. Tuttavia, una sfida ancora aperta è la diversità di linguaggio. In particolare, l'applicazione degli LLM per dialetti e lingue con risorse limitate rimane un ambito relativamente poco esplorato.

L'idea di base è quella di sfruttare i Large Language Model per progettare e sviluppare soluzioni che coinvolgano una lingua non standard. Le soluzioni pensate sono un chatbot, o assistente vocale, in grado di comprendere e interagire in un dialetto, in particolare il dialetto fermano.

Un chatbot può essere uno strumento interattivo per i turisti, in grado di rispondere a domande in dialetto e di spiegare tradizioni locali, punti d'interesse ed eventi culturali.

Un assistente vocale capace di interagire in dialetto potrebbe facilitare la comunicazione con utenti anziani o persone meno abituate all'italiano standard, migliorando l'accessibilità ai servizi attraverso una tecnologia che rispecchi la lingua parlata. L'introduzione di un chatbot di questo tipo potrebbe non solo attrarre l'interesse dei giovani, ma anche favorire il dialogo intergenerazionale, coinvolgendo le persone anziane che parlano quotidianamente il dialetto.

L'obiettivo è quello di preservare e valorizzare il dialetto fermano, contribuendo, al contempo, a sviluppare applicazioni pratiche nel turismo, nella cultura locale e nella comunicazione quotidiana.

I dialetti, così come le lingue minoritarie, non solo riflettono differenze regionali ma anche pragmatiche, derivanti da fattori sociali come la classe, l'etnia, il genere e l'età, che influenzano il modo in cui il linguaggio viene usato e interpretato.

Un modello linguistico in grado di comprendere e generare testo in dialetto fermano non è solo uno strumento per preservare una specifica tradizione linguistica, ma rappresenta

anche un esempio di come queste tecnologie possano essere applicate a lingue o varianti linguistiche con risorse limitate. Un modello in grado di comprendere e generare testo in più lingue può contribuire a migliorare la qualità della vita delle comunità linguistiche meno rappresentate, garantendo loro un accesso più equo a informazioni e servizi essenziali. Questo approccio può essere applicato tanto a dialetti locali quanto a lingue regionali, come quelle africane (es. Swahili, Yoruba, Hausa), promuovendo uno sviluppo inclusivo che valorizza le tradizioni linguistiche e culturali e amplia le possibilità di sviluppo tecnologico e di accesso alla conoscenza.

2.2 Raccolta dei requisiti

Dopo aver definito gli obiettivi e l'idea generale del progetto, è fondamentale identificare gli strumenti e i vari componenti necessari per la progettazione e lo sviluppo. Questa fase prevede la raccolta e la categorizzazione dei requisiti in funzionali e non funzionali per garantire che il sistema risponda sia alle esigenze di base sia alle aspettative in termini di qualità e prestazioni.

I requisiti funzionali descrivono le capacità e le funzionalità specifiche che il chatbot, o assistente vocale, dovrà possedere per interagire efficacemente in dialetto fermano. Essi includono:

- *Comprensione del dialetto*: il sistema deve essere in grado di interpretare e comprendere correttamente le domande e le affermazioni in dialetto fermano. Esso deve essere in grado di interpretare non solo le parole, ma anche gli accenti specifici del dialetto fermano. Questo requisito richiede che il modello sia addestrato su un corpus di dati in dialetto, comprendendo le peculiarità lessicali, sintattiche e foniche del fermano.

- *Generazione di risposte coerenti*: il chatbot deve saper generare risposte fluenti e appropriate in dialetto, rispettando le particolarità del linguaggio parlato locale.

- *Riproduzione dell'accento dialettale*: l'assistente deve essere in grado di generare risposte con l'intonazione e l'accento tipici del dialetto fermano, per creare un'interazione più naturale e autentica per gli utenti locali.

- *Personalizzazione del contesto*: il modello deve essere capace di adattarsi al contesto della conversazione, fornendo risposte rilevanti e coerenti a seconda del flusso del dialogo e dell'argomento trattato, come informazioni su eventi locali, tradizioni, o luoghi turistici.

- *Interazione multi-turno*: il sistema dovrebbe gestire interazioni multi-turno, mantenendo la continuità del discorso per migliorare l'esperienza utente, in modo da seguire correttamente una conversazione composta da più domande e risposte.

I requisiti non funzionali definiscono le caratteristiche di qualità del sistema, che ne influenzano l'efficacia, l'affidabilità e l'esperienza d'uso. Questi includono:

- *Accuratezza*: il chatbot deve garantire un'elevata accuratezza nella comprensione e generazione del dialetto fermano. È fondamentale che il sistema rispetti le peculiarità linguistiche locali senza compromettere la qualità delle risposte.

- *Prestazioni e tempi di risposta*: il sistema deve essere in grado di rispondere rapidamente alle domande degli utenti, minimizzando i tempi di latenza, per garantire un'interazione fluida e in tempo reale.

- *Scalabilità*: il modello dovrebbe essere progettato in modo da poter gestire un numero crescente di richieste e interazioni senza degradare le prestazioni. Questo è particolarmente importante se l'applicazione viene implementata su larga scala, ad esempio in un contesto turistico.

- *Robustezza e flessibilità*: il chatbot deve essere robusto e in grado di gestire variazioni e incertezze nel linguaggio dell'utente, incluse abbreviazioni, variazioni dialettali e rumori di fondo.

- *Privacy e sicurezza dei Dati*: poichè il sistema può raccogliere dati personali durante le interazioni, è necessario implementare misure per proteggere la privacy degli utenti e garantire che tutte le informazioni vengano gestite in conformità con le normative di protezione dei dati.

Per realizzare un modello in grado di comprendere e generare testo in dialetto fermano sono necessari diversi strumenti. Di seguito sono elencati i componenti principali:

- *Corpus dialettale*: è essenziale disporre di un dataset ampio e rappresentativo in dialetto fermano. Questo può includere trascrizioni di dialoghi, testi dialettali, interviste o materiale audio da cui estrarre testi rappresentativi del dialetto. Un corpus adeguato consente al modello di apprendere le peculiarità linguistiche specifiche del dialetto.

- *Dataset misto*: è necessario utilizzare anche dati in italiano standard e in altri dialetti italiani per rafforzare le capacità generative del modello e facilitarne la comprensione del contesto linguistico. Questo approccio può aiutare il modello a migliorare la traduzione tra italiano e dialetto, oltre ad ampliare la capacità di comprensione.

- *Corpus del parlato dialettale*: è fondamentale disporre di un dataset di parlato in dialetto fermano, con registrazioni audio che includano varie voci e inflessioni tipiche. Questo dataset servirà per addestrare il modello non solo a comprendere il testo dialettale, ma anche a riconoscere e generare l'accento specifico, riproducendo l'intonazione e le sfumature della lingua parlata locale.

- *Modelli pre-addestrati*: è necessario utilizzare modelli come BERT, GPT o Minerva come base su cui costruire il chatbot. Questi modelli hanno già appreso le strutture linguistiche fondamentali attraverso dataset di grandi dimensioni, e possono essere successivamente adattati al dialetto fermano tramite fine-tuning.

- *Modelli dialettali specifici*: l'impiego di modelli già addestrati su altri dialetti italiani può fornire un vantaggio per il fine-tuning sul dialetto fermano, velocizzando l'addestramento e migliorando la capacità del modello di generalizzare su lingue minoritarie o regionali.

- *Modelli di sintesi vocale*: per la generazione di un output vocale con un accento dialettale, potrebbe essere utile adottare modelli di sintesi vocale, come TTS (Text-to-Speech), che possano riprodurre fedelmente l'intonazione del dialetto.

- *Infrastruttura computazionale*: disporre di risorse computazionali adeguate, come GPU o TPU, è fondamentale per addestrare e ottimizzare un modello di linguaggio su dataset dialettali, che richiedono elevate capacità di calcolo per processare e apprendere efficacemente il testo.

- *Tecniche di Fine-Tuning*: il fine-tuning su corpus dialettali permette di specializzare il modello, migliorando la sua accuratezza e precisione nel comprendere e generare testi

in dialetto fermano. Questo processo richiede l'uso di framework di machine learning, come PyTorch o TensorFlow, e librerie specifiche, come Hugging Face Transformers, per semplificare l'implementazione e la gestione dei modelli.

2.3 Studio di fattibilità

Lo studio di fattibilità consiste nell'analisi e nella valutazione delle limitazioni tecniche e pratiche e delle risorse disponibili per l'implementazione dell'idea generale.

Il principale ostacolo è la disponibilità limitata di testi e risorse in dialetto fermano. La raccolta di un corpus dialettale richiede tempo e può implicare la necessità di trascrivere e annotare manualmente conversazioni. Il dialetto fermano, come molti altri dialetti e lingue, presenta variabilità fonetiche e grammaticali che possono essere difficili da modellare, specialmente se le risorse linguistiche adeguate sono scarse. Inoltre, l'implementazione di un assistente vocale che comprenda e risponda in dialetto fermano potrebbe richiedere la creazione di modelli di riconoscimento vocale specifici per il dialetto, una sfida ulteriore data la complessità del parlato locale.

Per quanto riguarda le limitazioni pratiche, l'addestramento di un Large Language Model su dialetti richiede risorse computazionali elevate, come GPU o TPU, non sempre sufficienti.

Una sfida da affrontare sta anche nella valutazione qualitativa del testo generato dal modello. Per garantire l'accuratezza linguistica, potrebbe essere utile collaborare con esperti del dialetto fermano o parlanti nativi che possano validare e correggere i dati raccolti e i risultati prodotti.

Infine, l'implementazione di un modello efficiente richiede un periodo di sviluppo e testing significativo, soprattutto nella fase di raccolta e pre-processing del dataset.

Alla luce delle risorse disponibili, l'idea iniziale del progetto è stata ridimensionata per focalizzarsi su obiettivi concreti e realizzabili nel contesto attuale. L'idea di un chatbot, o assistente vocale completo, in dialetto fermano rappresenta un progetto ambizioso, ma per limitazioni di tempo e risorse, si è deciso di concentrare il lavoro su una fase preliminare fondamentale: la raccolta e l'elaborazione dei dati per eseguire il fine-tuning di modelli linguistici esistenti e valutare le loro performance.

In particolare, i dati raccolti sono stati utilizzati per effettuare il fine-tuning su alcuni Large Language Model, adattandoli alle specificità del dialetto fermano e testando la loro capacità di comprensione e generazione di testo in un contesto dialettale. Questa fase rappresenta un primo passo essenziale verso lo sviluppo di un sistema più completo e contribuisce a porre le basi per eventuali miglioramenti futuri.

CAPITOLO 3

Implementazione

In questo capitolo si approfondiranno le varie fasi implementate per effettuare il fine-tuning di alcuni Large Language Model sul dialetto fermano. Il primo step descritto sarà la raccolta dei dati necessari per l'addestramento, quindi la loro estrazione e pulizia. Successivamente verranno illustrate le tecniche di pre processing del dataset per consentire l'addestramento. Verranno approfondite le scelte di modeling per quanto riguarda i modelli più adatti ai task definiti. La fase principale di fine-tuning verrà descritta in modo accurato specificando le procedure relative ai diversi task. Infine, verranno discusse le metriche di valutazione adottate per misurare le performance del modello.

3.1 Acquisizione dati

L'acquisizione dei dati rappresenta il primo step in una pipeline di Natural Language Processing (NLP). Questo passaggio è fondamentale poiché i dati costituiscono il fulcro di qualsiasi modello di machine learning o deep learning; la quantità e la qualità dei dati sono fattori essenziali per garantire un addestramento efficace.

Nel contesto di un modello dialettale, l'attenzione alla raccolta di dati di qualità è ancora più cruciale, in quanto non esistono grandi quantità di risorse linguistiche standardizzate e facilmente accessibili per queste varianti locali. È fondamentale, quindi, che i dati raccolti siano rappresentativi cioè rispecchino fedelmente il vocabolario, le strutture sintattiche e le espressioni idiomatiche tipiche del dialetto.

La raccolta dei dati si è basata su consultazioni online e indagini presso biblioteche e archivi locali. Essendo il dialetto fermano una lingua minoritaria, non esistono corpora digitali standardizzati o database di riferimento come quelli disponibili per le lingue maggiori.

Attraverso queste ricerche sono stati individuati principalmente materiali sotto forma di poesie e commedie dialettali.

Di seguito è riportato un elenco di alcuni dei libri utilizzati per costruire il dataset:

- *La luna mi raccontò* – Filippo Cruciani, 2007.
- *La vita è 'na commedia* - Gabriele Mancini, 2014 (Figura 3.1).
- *Tavole di palcoscenico* - Massimo Mezzanotte, 1999.
- *'Ntunì e la sua terra, Volume I* - Antonio Angelelli, 1996 (Figura 3.2).
- *'Ntunì e la sua terra, Volume II* - Antonio Angelelli, 1996.

CHI E' PIU' LESTI...SE LA GUADAGNA!

Peppino: Prima de tutto te vulìo chiede…visto che orma' scimo 'rriàti a 'stu pundu…

Antò: Non me devi chiede probbio gnende, te lo sci pijato da sulu quello che vulii, adde' che cerchi..!

Peppino: Io te vulìo chiede…se te pòzzo chiama'…babbo…

Antò: Io te dico che….comme sci 'itto..?

Peppino: Te so' ditto…se te pòzzo chiamà babbo!…perché io e Giovannetta ce vulìmo vène e ce vulìmo sposa'..!

Antò: E ce vulìa tando? Se lo e ravate 'itto de prima non era mèjo..?

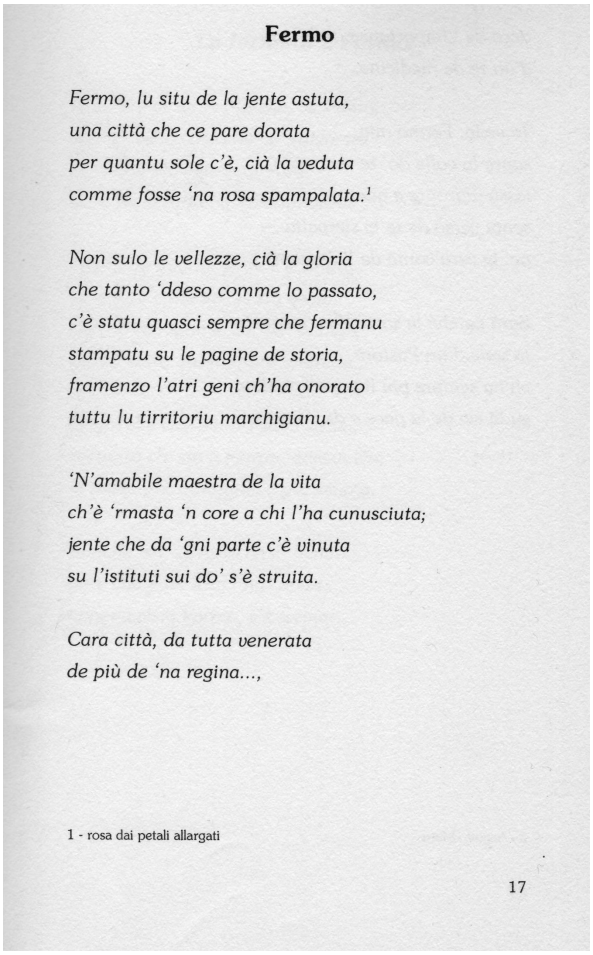
Giovannetta: Forse ciài ragio'…ma pensavamo che vuàtri non eravate daccórdu e allora…

319

Figura 3.1: Scansione di una pagina del libro "La vita è 'na commedia"

Questi generi letterari sono particolarmente utili, in quanto conservano espressioni e modi di dire tipici. Le commedie dialettali hanno permesso di raccogliere dialoghi in contesti reali, come discussioni tra familiari, interazioni sociali, e racconti di esperienze quotidiane, tutti elementi cruciali per un chatbot che sappia rispondere in modo naturale.

L'uso di fonti come poesie e commedie introduce, però, un rischio di bias sociali e culturali. Questi testi, inoltre, non coprono ampie tematiche di cultura generale o conoscenze scientifiche.



Fermo

Fermo, lu situ de la jente astuta,
una città che ce pare dorata
per quantu sole c'è, cià la veduta
comme fosse 'na rosa spampalata.[1]

Non sulo le vellezze, cià la gloria
che tanto 'ddeso comme lo passato,
c'è statu quasci sempre che fermanu
stampatu su le pagine de storia,
framenzo l'atri geni ch'ha onorato
tuttu lu tirritoriu marchigianu.

'N'amabile maestra de la vita
ch'è 'rmasta 'n core a chi l'ha cunusciuta;
jente che da 'gni parte c'è vinuta
su l'istituti sui do' s'è struita.

Cara città, da tutta venerata
de più de 'na regina...,

1 - rosa dai petali allargati

17

Figura 3.2: Scansione di una pagina del libro "'Ntunì e la sua terra, Volume I"

Il fine-tuning su dataset con questo contenuto comporta il rischio di catastrophic forgetting, un fenomeno che si verifica quando un modello addestrato su nuovi dati perde progressivamente le conoscenze generali apprese in precedenza. Il modello potrebbe, quindi, non rispondere correttamente a domande di cultura generale, storia, scienza o argomenti accademici che non compaiono nei dati di addestramento.

3.2 Estrazione e pulizia dati

Il secondo step della pipeline riguarda la digitalizzazione e una prima pulizia dei dati, un passaggio fondamentale per garantire che il dataset finale sia adatto all'addestramento del modello.

Data la natura cartacea delle fonti originarie, è stato necessario convertire questi documenti in formato digitale attraverso un processo di scansione e riconoscimento ottico dei caratteri (OCR), seguito da un'attenta pulizia manuale dei dati estratti.

Il processo di digitalizzazione ha coinvolto due fasi principali:

1. *Scansione delle pagine*: ogni documento cartaceo è stato scansionato pagina per pagina.
2. *Conversione in testo*: successivamente, le immagini ottenute sono state elaborate dal software *Image Scan OCR*, che ha estratto il testo dalle immagini e lo ha convertito in formato digitale.

Nella Figura 3.3 è riportato uno screenshot dell'applicazione per l'OCR a partire dalla scansione di una pagina di uno dei libri usati per la costruzione del dataset.

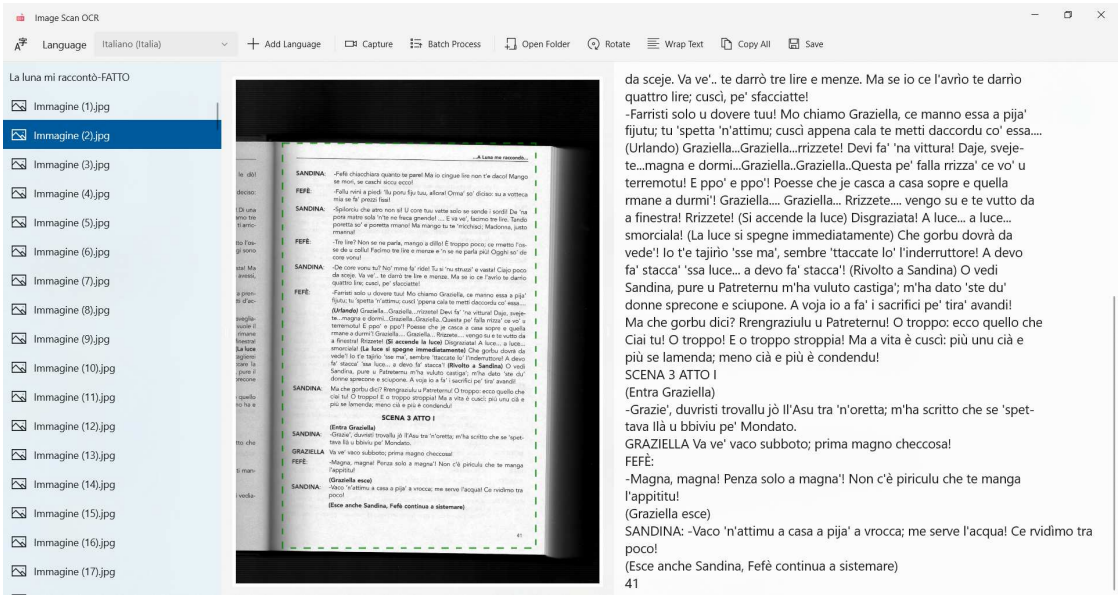


Figura 3.3: Screenshot del software OCR

L'uso del software OCR ha permesso di ottenere una digitalizzazione accurata dei testi ma, come spesso accade, il processo OCR ha generato alcuni errori tipici, quali riconoscimenti errati di caratteri o spaziature irregolari. Per questo motivo si è resa necessaria una fase successiva di pulizia dei dati per garantire la qualità del testo estratto.

È stata, quindi, condotta una prima pulizia manuale del testo per eliminare elementi superflui e preparare i dati per l'addestramento del modello. Questa fase di pulizia ha incluso le seguenti attività:

- *Rimozione dei numeri di pagina e intestazioni*: dato che i documenti originali contenevano numerazioni di pagina e intestazioni che si ripetevano, queste sono state rimosse manualmente dal testo. Questi elementi sono estranei al contenuto utile per l'addestramento e potrebbero introdurre rumore nel dataset.

- *Eliminazione delle indicazioni sceniche*: poiché il materiale raccolto includeva indicazioni per gli attori, come "si alza in piedi" o "si accende la luce," è stato necessario eliminare manualmente tali annotazioni. Questi dettagli, essendo parte delle istruzioni di scena e non del testo di dialogo, avrebbero potuto confondere il modello durante l'addestramento, influendo negativamente sulla comprensione linguistica.

- *Correzione degli errori OCR*: infine, sono stati rivisti e corretti alcuni errori di riconoscimento tipici del processo OCR. Questi errori riguardavano spesso lettere simili, come la confusione tra "l" e "I" o tra "0" e "O." La correzione di questi errori è fondamentale per mantenere la qualità e la coerenza del dataset finale.

Nella Figura 3.4 viene mostrato un esempio del testo in formato PDF dopo questa prima fase di pulizia manuale.

Al termine della pulizia dei dati, il testo risultante è stato organizzato in un documento PDF, pronto per le fasi successive di pre-processing e addestramento del modello.

L'accuratezza e la qualità dei dati sono fondamentali per il successo dell'addestramento di un Large Language Model. Ogni passaggio di pulizia e verifica contribuisce a migliorare la rappresentatività e la coerenza del dataset.

-Farristi solo u dovere tuu! Mo chiamo Graziella, ce manno essa a pija' fijutu; tu 'spetta 'n'attimu; cuscì appena cala te metti daccordu co' essa.... Graziella...Graziella...rrizzete! Devi fa' 'na vittura! Daje, svejete...magna e dormi...Graziella..Graziella..Questa pe' falla rrizza' ce vo' u terremotu! E ppo' e ppo'! Poesse che je casca a casa sopre e quella rmane a durmi'! Graziella.... Graziella... Rrizzete.... vengo su e te vutto da a finestra! Rrizzete! Disgraziata! A luce... a luce... smorciala! Che gorbu dovrà da vede'! Io t'e tajirìo 'sse ma', sembre 'ttaccate lo' l'inderruttore! A devo fa' stacca' 'ssa luce... a devo fa' stacca'! O vedi Sandina, pure u Patreternu m'ha vuluto castiga'; m'ha dato 'ste du' donne sprecone e sciupone. A voja io a fa' i sacrifici pe' tira' avandi!

Ma che gorbu dici? Rrengraziulu u Patreternu! O troppo: ecco quello che Ciai tu! O troppo! E o troppo stroppia! Ma a vita è cuscì: più unu cià e più se lamenda; meno cià e più è condendu!

-Grazie', duvristi trovallu jò Il'Asu tra 'n'oretta; m'ha scritto che se 'spet- tava Illà u bbiviu pe' Mondato.

GRAZIELLA Va ve' vaco subboto; prima magno checcosa!

FEFÈ: -Magna, magna! Penza solo a magna'! Non c'è piriculu che te manga l'appititu!

SANDINA: -Vaco 'n'attimu a casa a pija' a vrocca; me serve l'acqua! Ce rvidimo tra poco!

-Comba' Fefè!

-O comma' Righetta! Comme mai cuscì presto?

-So' vinuta a pija' l'acqua. O sai, è justo che 'na matre de famija se rrizza presto a matina. C'è tando da fa' : mica se po' sta' a llillasse dendro u lettu! E ppo' io no' m'o pozzo permette!

-O saccio, o saccio. Chi dorme non pija pesci!

-E ppo', a casa mia, 'ppena canda u gallu, tutti su, prima de me!

-O saccio, o saccio!

-Pure a domeneca; tutti in piedi presto ! Tutti a messa prima! A jende non cià probbio gnende da di' su a famija nostra!.....Piuttosto a commarella Graziella s'è già rrizzata?

-Certo! E ppo' da tando! E jita a fa' 'na vittura!

-Quand'è brava 'lla commarella mia! Vrava e bella! Devi esse condendu d'ave' 'na fija cuscì!

Figura 3.4: Esempio di testo dopo la prima pulizia

La rimozione delle informazioni superflue e la correzione degli errori di trascrizione consentono al modello di concentrarsi sui pattern linguistici e sulle caratteristiche del testo, senza essere influenzato da informazioni irrilevanti o errori che potrebbero compromettere i risultati.

3.3 Pre-Processing

Il Pre-Processing rappresenta una fase fondamentale per preparare i dati in un formato coerente e utilizzabile dal modello.

La pulizia manuale iniziale del testo non è stata sufficiente a garantire un dataset completamente pulito e pronto per l'addestramento del modello. Dopo la digitalizzazione e una prima revisione, è emersa la necessità di sviluppare uno script Python per automatizzare le operazioni di pre-processing. Questo script ha permesso di applicare diverse funzioni di pulizia e trasformazione del testo in modo efficiente e su larga scala, garantendo un output finale più coerente e strutturato.

A seguito dell'estrazione del testo dal formato cartaceo, è stata creata una funzione (Listato 3.1) per estrarre il testo dai vari PDF facilitando l'elaborazione successiva.

```
1 def extract_text_from_pdf(pdf_path):
2     pdf_document = fitz.open(pdf_path)
3     text = ""
4     for page_num in range(len(pdf_document)):
5         page = pdf_document.load_page(page_num)
6         text += page.get_text()
7     return text
```

Listato 3.1: Funzione per estrarre il testo dai PDF

Dopo di ciò, sono state implementate diverse funzioni di pulizia per rimuovere gli elementi non necessari e ottenere un testo accurato e uniforme.

Utilizzando espressioni regolari, è stato eliminato ogni riferimento numerico (come numeri di pagina o apici per le note), che non sarebbe stato utile per l'addestramento. Questa operazione include anche la sostituzione dei trattini (che rappresentano l'inizio di una battuta della commedia) con spazi e la rimozione di simboli non alfabetici, ad eccezione dei segni di punteggiatura rilevanti. La funzione in questione è riportata nel Listato 3.2.

```
def clean_text(text):
    # Rimuove tutti i numeri
    text = re.sub(r'\d+', '', text)
    # Sostituisce i trattini con spazi
    text = text.replace('-', ' ')
    # Rimuove tutti i simboli non alfabetici o di spazio
    text = re.sub(r'[^\w\s.,!?\'']', '', text)
    # Rimuove numeri in apice
    text = re.sub(r'[0123456789]', '', text)
    # Rimuove i ritorni a capo non preceduti da ., !, ?
    text = re.sub(r'(?<![.!?])\n+', ' ', text)
    # Mantiene i ritorni a capo preceduti da ., !, ?
    text = re.sub(r'([.!?])\s*\n', r'\1\n', text)
    # Rimuove spazi bianchi multipli
    text = re.sub(r'\s+', ' ', text)
    # Rimuove spazi bianchi iniziali e finali
    return text.strip()
```

Listato 3.2: Funzione per pulire il testo estratto

Analizzando i dataset, soprattutto per quanto riguarda le commedie dialettali, è stato notato che ad ogni inizio battuta è riportato il nome del personaggio incaricato a recitare la frase. Per evitare che il modello apprenda correlazioni non necessarie legate ai nomi dei personaggi, è stata creata una funzione per rimuovere specifici nomi dal testo. La funzione nel Listato 3.3 elimina ogni nome della lista creata manualmente, ignorando differenze di maiuscole e minuscole e rispettando la formattazione del testo. Questo passaggio aiuta a eliminare elementi di disturbo e a rendere il dataset meno rumoroso.

```
def remove_name(text, name_list):
    for name in name_list:
        text = re.sub(r"([\'']?)\b" + re.escape(name) + r"\b([\'']?)", '', text, flags=re.IGNORECASE)
    return text
```

Listato 3.3: Funzione per rimuovere i nomi dal testo

Un passaggio fondamentale nel pre-processing è la segmentazione, cioè la suddivisione del testo in frasi. La segmentazione è stata eseguita tramite una funzione specifica che identifica i segni di punteggiatura principali (punto, punto esclamativo e punto interrogativo) e aggiunge un'interruzione di riga dopo ciascun segno, separando, così, il testo in frasi complete. Questo passaggio è particolarmente utile per i modelli linguistici, poiché consente ad essi di apprendere in modo più efficace le relazioni tra parole e frasi. Sono stati, inoltre, creati metodi di segmentazione personalizzati per gestire vari tipi di documenti, come libri e poesie. Ad esempio, per alcuni documenti, il trattino è stato sostituito con un'interruzione di riga, mentre in altri casi (ad esempio, nelle poesie) si è proceduto ad aggiungere interruzioni di riga dopo ogni punto, per assicurare che il testo fosse segmentato correttamente.

Un'altra fase indispensabile del pre-processing consiste nella tokenizzazione, ovvero nella suddivisione delle frasi in parole o token. Questa operazione è essenziale per la preparazione dei dati, poiché rende le informazioni linguistiche comprensibili al modello. Tuttavia, la tokenizzazione è specifica per ogni modello utilizzato e richiede parametri personalizzati, come il tipo di tokenizer o il task da eseguire. Pertanto, la descrizione dettagliata di questa fase sarà fornita nella sezione successiva, dove saranno illustrati i passaggi e gli strumenti specifici adottati per ciascun Large Language Model impiegato.

Le funzioni di pulizia del testo e segmentazione sono state applicate singolarmente a ciascun file PDF raccolto durante la fase preliminare. Ogni documento, rappresentante uno specifico libro o testo di interesse, è stato elaborato per rimuovere elementi non rilevanti, normalizzare il formato e segmentare il contenuto in frasi. Al termine di questo processo, i testi puliti e uniformati sono stati uniti in un unico file `.txt`, che costituisce la base per le

fasi successive di elaborazione. Una porzione di esempio del dataset finale è riportata nella Figura 3.5.

```
'sta òta è statu grossu!
lu sci sintitu lu terremotu?
Lu sci sintitu se è statu grossu?
 Vona , statte vona!
Ormai è passatu, statte calma e mettete a sedè!
 Stavo a sistemà li sòrdi in un postu sicuru quanno ha 'ncuminciato a ballà tutto.
oddio se ce rpenzo!
 Eccheme ecco, , so' fatto tutto.
 E fermete, refiata.
pare che te scia vinutu 'rreto un ca' guastù.
 Pègghio, , pègghio!
Ma no lu scì sintitu se è statu grossu?
Vallava tutto.
le case paria che me vulìa cascà sopre, li pali de la luce java nenquà e nellà.
Scine paria proprio lu finimunnu!
 E lascia perde 'sso matto!
Piuttosto, sci portato tutto lo necessario?
 Scìne , sò pijato tutto quello che putìa sirvì pe' fa 'na gabbia.
 Famme mpô vede se per daero ce sta tutto, perché de te ce sta poco da fidas se e io vurrìo preparamme vène tutta l'attrezzatura, cusci
pozzo 'ncumincià!
 Scìne Nico, scìne.
è mejo che te sbrighi, prima lu finisci lu lauro e mejo è!
Li sordi ce farrìa commudu dopràlli subboto.
 Sapete jente, io ci avrìo 'na spesetta justa justa da fa'!
 E tu che cce rrentri?
 Comme che cce rrentro?
Non ve staco jutenno?
```

Figura 3.5: Esempio di testo dopo il pre-processing

3.4 Modeling

Il modeling nel contesto del Natural Language Processing si riferisce alla selezione, alla configurazione e all'addestramento di modelli di linguaggio per risolvere compiti specifici, come la generazione di testo, la classificazione o la comprensione del linguaggio naturale. In questa fase vengono scelti i modelli più adatti alle esigenze e ai requisiti, ottimizzandone le prestazioni tramite il fine-tuning su un dataset mirato. Il modeling rappresenta quindi una fase cruciale per trasformare un modello pre-addestrato in uno strumento efficace per il task specifico.

3.4.1 Definizione dei task

I Large Language Model sono stati configurati per affrontare tre task principali:

- *Generazione di testo*: questo task consiste nella capacità del modello di produrre testo coerente e contestuale a partire da un prompt iniziale. Il modello deve comprendere il contesto dato e continuare la frase, applicando conoscenze strutturali e semantiche per mantenere un flusso naturale e comprensibile. La generazione di testo è particolarmente rilevante per un chatbot, o assistente vocale, in dialetto, poiché consente di produrre risposte fluide e di interagire in modo naturale con l'utente.

- *Target Word Prediction (TWP)*: in questo task, una parola all'interno di una frase è mascherata e il modello deve predire la parola mancante basandosi sul contesto circostante. Questo task valuta la capacità del modello di cogliere le relazioni tra le parole e di prevedere il termine corretto in base alle informazioni fornite nel resto della frase. Per esempio, dato un contesto parziale, il modello dovrebbe poter intuire la parola corretta che completa il significato della frase.

- *Target Word Selection (TWS)*: a differenza della TWP, il task di Target Word Selection presenta al modello una frase troncata insieme a una lista di possibili parole candidate

e il modello deve selezionare il token che completa correttamente la frase. Questo task richiede al modello non solo di riconoscere il contesto, ma anche di distinguere tra opzioni simili e di identificare la parola più appropriata per completare la frase in modo semantico e grammaticale. Il TWS è un task utile per valutare la capacità del modello di differenziare sfumature linguistiche e sintattiche tra parole simili.

La definizione di questi task guida la selezione dei modelli e determina il tipo di preprocessing dei dati necessario per adattarli ai vari contesti applicativi.

3.4.2 Scelta dei modelli

Nel contesto in questione, la scelta dei modelli per la valutazione delle performance sul dataset dialettale e il confronto è stata orientata verso architetture consolidate per l'italiano; in particolare, sono stati considerati Minerva (*"sapienzanlp/Minerva-350M-base-v1.0"*) e BERT (*"dbmdz/bert-base-italian-uncased"*).

Minerva è la prima famiglia di Large Language Model pre-addestrata interamente in italiano. Si tratta di un modello di linguaggio sviluppato da Sapienza NLP in collaborazione con Future Artificial Intelligence Research (FAIR) e CINECA per rispondere alle esigenze di comprensione e generazione della lingua italiana, con particolare attenzione alla generazione del testo. Minerva si basa sull'architettura GPT-like, un modello di tipo Transformer che si distingue per la struttura decoder-only, dove l'obiettivo è generare testo predittivamente, parola per parola, basandosi sul contesto precedente. L'approccio è di tipo autoregressivo, ovvero il modello genera il testo sequenzialmente, utilizzando i token precedenti per prevedere il successivo. Grazie a questo meccanismo, Minerva si adatta molto bene ai task di generazione e completamento del testo.

La scelta di Minerva è motivata dalla sua capacità di produrre testo coerente e contestuale in lingua italiana, caratteristica che lo rende particolarmente adeguato ad adattarsi alla sintassi e alle peculiarità linguistiche del dialetto fermano. Inoltre, esso risulta perfettamente allineato con l'obiettivo di sviluppare un chatbot in grado di generare testo in dialetto, sfruttando la sua capacità di adattamento a contesti linguistici specifici.

BERT (Bidirectional Encoder Representations from Transformers) è una delle architetture più influenti nel panorama dell' NLP. Sviluppato da Google, BERT si basa su un'architettura encoder-only e opera in modo bidirezionale cioè cerca di comprendere il significato delle parole analizzando il contesto dell'intera frase (e non solo delle parole viste in precedenza). Si tratta di un modello addestrato su due task principali: il Masked Language Modeling (MLM) e il Next Sentence Prediction (NSP). L'MLM consiste nel mascherare a caso il 15% delle parole in ingresso, sostituendole con un token [MASK], far passare l'intera sequenza attraverso l'encoder basato sulla self-attention di BERT, e quindi prevedere solo le parole mascherate, basandosi sul contesto fornito dalle altre parole non mascherate della sequenza. Nell'NSP durante l'addestramento il modello viene alimentato con due frasi in ingresso alla volta; BERT deve prevedere se la seconda frase è casuale o correlata contestualmente alla prima. Questo meccanismo rende BERT particolarmente adatto a compiti di comprensione del linguaggio, come il question answering e la sentiment analysis, dove il modello deve interpretare il contesto completo della frase.

La versione di BERT scelta, *"dbmdz/bert-base-italian-uncased"*, è stata addestrata specificamente in italiano, facilitando il fine tuning sul dataset dialettale. A differenza di Minerva, BERT non è concepito per la generazione del testo ma eccelle nella comprensione e classificazione, permettendo, quindi, un'analisi approfondita delle strutture testuali. Questo modello è stato selezionato per eseguire un confronto con Minerva nella comprensione linguistica e per analizzare il comportamento dei diversi modelli su task differenti.

In particolare, Minerva è stato selezionato per le sue capacità di generazione di testo, mentre BERT è stato scelto per la sua capacità di comprendere e interpretare il contesto delle frasi, il che lo rende adatto per i task di TWP e TWS.

Sono stati valutati altri modelli di ultima generazione, come LLaMA (Large Language Model Meta AI) e GPT-3, ma non sono stati selezionati per ragioni specifiche. LLaMA, pur essendo un modello potente e altamente performante, richiede una capacità computazionale molto elevata e risorse che possono risultare impegnative per implementazioni focalizzate su dialetti specifici, come il fermano. Inoltre, GPT-3 e altri modelli di linguaggio di grandi dimensioni presentano limitazioni legate all'adattabilità a lingue minoritarie e dialetti, in quanto l'addestramento su ampie basi di dati multilingue non include necessariamente varietà linguistiche specifiche.

Pertanto, la scelta di Minerva e BERT rappresenta un bilanciamento tra efficacia, adattabilità e sostenibilità, garantendo prestazioni adeguate senza richiedere risorse computazionali eccessive.

3.4.3 Tokenizzazione

Come spiegato nelle sezioni precedenti, per la fase di addestramento, è stato innanzitutto necessario preparare il dataset attraverso un processo di tokenizzazione. La tokenizzazione è il passaggio in cui il testo viene suddiviso in unità minime, dette token, che possono essere parole, sottoparole o caratteri, a seconda del tokenizer adottato.

Per tale scopo, è stata utilizzata la funzione `AutoTokenizer.from_pretrained` di Hugging Face che consente di caricare automaticamente un tokenizer pre-addestrato direttamente dalla libreria online di modelli pre-addestrati. Questa funzione è parte della classe *AutoTokenizer*, che suddivide il testo in token secondo le regole definite dal tokenizer pre-addestrato specifico del modello.

L'implementazione della tokenizzazione del dataset, nel caso di Minerva per il task di generazione testo e TWP, è mostrata nel Listato 3.4; per quanto riguarda Bert per il TWP, la procedura è analoga ad eccezione del nome del modello specificato nell'*AutoTokenizer*.

Nello specifico i passi eseguiti sono:

1. *Caricamento e filtraggio del dataset*: è stato caricato il dataset dal file `.txt` precedentemente pulito e sono stati filtrati i dati in modo da mantenere solo i testi contenenti almeno tre parole, assicurando che ogni input fornisca un minimo contesto linguistico.

2. *Shuffling del dataset*: il dataset è stato poi mischiato (randomizzato) con un seed fisso per garantire la riproducibilità.

3. *Tokenizzazione*: il tokenizer è stato configurato per eseguire il padding (aggiunta di simboli di riempimento per uniformare la lunghezza delle sequenze) e il truncation (taglio delle sequenze troppo lunghe) con una lunghezza massima di 128 token per sequenza. La *tokenize_function* viene applicata in batch sull'intero dataset, cioè su blocchi di dati contemporaneamente. Questo approccio è più efficiente rispetto all'elaborazione di una frase alla volta e sfrutta meglio le capacità computazionali, riducendo il tempo necessario per la tokenizzazione.

```
# Carica il dataset
dataset = load_dataset('text', data_files={'train': 'dataset_processed.txt'})
# Filtra il dataset. Viene mantenuto solo il testo che contiene almeno 3 parole
dataset = dataset.filter(lambda example: len(example['text'].split()) >= 3)
# Shuffling del dataset
dataset = dataset.shuffle(seed=42)

# Carica il tokenizer pre-addestrato
tokenizer = AutoTokenizer.from_pretrained("sapienzanlp/Minerva-350M-base-v1.0")

```

```
# Imposta il token di padding (pad_token) uguale al token di fine sequenza (eos_token)
tokenizer.pad_token = tokenizer.eos_token

# Tokenizzazione del dataset
def tokenize_function(examples):
    return tokenizer(
        examples['text'],
        truncation=True,  #Tronca le frasi lunghe
        padding="max_length",
        max_length=128,
    )

tokenized_datasets = dataset.map(tokenize_function, batched=True)
```

Listato 3.4: Tokenizzazione del dataset per TWP

Nel task di Target Word Selection (TWS), è stato necessario un pre-processing specifico per creare un dataset di tipo *Multiple Choice*, in cui il modello ha accesso a una lista di parole candidate tra cui deve selezionare quella corretta in base al contesto della frase. L'implementazione è mostrata nel Listato 3.5.

```
# Funzione per estrarre l'ultima parola di una frase
def extract_last_word(sentence):
    words = sentence.split()
    return words[-1]

# Creazione di una lista di parole candidate, includendo la parola corretta
def create_candidate_words(sentence, dataset_sentences, num_candidates=4):
    correct_word = extract_last_word(sentence)
    all_words = [extract_last_word(s) for s in dataset_sentences if s != sentence]
    candidates = random.sample(all_words, num_candidates - 1)
    candidates.append(correct_word)
    random.shuffle(candidates)
    return candidates

# Funzione per generare il dataset Multiple Choice con labels
def preprocess_function(examples):
    questions = []
    options = []
    labels = []
    correct_words = []

    for example in examples['text']:
        sentence = example
        truncated_sentence = ' '.join(sentence.split()[:-1])  # Frase troncata senza l'ultima parola
        candidate_words = create_candidate_words(sentence, dataset['train']['text'])
        correct_word = extract_last_word(sentence)

        questions.append(truncated_sentence)
        options.append(candidate_words)
        labels.append(candidate_words.index(correct_word))
        correct_words.append(correct_word)

    # Tokenizza le domande e le opzioni
    first_sentences = [[q] * 4 for q in questions]  # Ripete la domanda per ciascuna opzione
    second_sentences = options  # Le opzioni come seconda parte

    # Appiattisce le liste per il tokenizer
    first_sentences = sum(first_sentences, [])
    second_sentences = sum(second_sentences, [])

    # Tokenizza
    tokenized_examples = tokenizer(first_sentences, second_sentences, truncation=True, padding=True)

    # Riorganizza per il task di Multiple Choice
    return {
        'input_ids': [tokenized_examples['input_ids'][i:i + 4] for i in range(0, len(tokenized_examples['
            input_ids']), 4)],
        'attention_mask': [tokenized_examples['attention_mask'][i:i + 4] for i in range(0, len(
            tokenized_examples['attention_mask']), 4)],
        'labels': labels,
        'options': options,
        'correct_word': correct_words,
    }

# Carica il tokenizer pre-addestrato
tokenizer = AutoTokenizer.from_pretrained("dbmdz/bert-base-italian-uncased")

# Carica il dataset
dataset = load_dataset('text', data_files={'train': 'dataset_processed.txt'})

# Filtra il dataset. Viene mantenuto solo il testo che contiene almeno 3 parole
dataset = dataset.filter(lambda example: len(example['text'].split()) >= 3)

# Shuffling del dataset
dataset = dataset.shuffle(seed=42)

# Applica la funzione di preprocessing al dataset
tokenized_datasets = dataset.map(preprocess_function, batched=True)
```

Listato 3.5: Tokenizzazione del dataset per TWS

La procedura si basa sulle seguenti fasi:

1. *Estrazione della parola corretta*: la funzione `extract_last_word` estrae l'ultima parola di ciascuna frase nel dataset. Questa parola rappresenta la "target word", o parola corretta che completa il contesto dato.

2. *Creazione delle parole candidate*: con la funzione `create_candidate_words`, per ciascuna frase, viene generato un elenco di parole candidate, di cui solo una è corretta. Per fare ciò, vengono selezionate casualmente altre parole finali provenienti dal dataset e si aggiunge la parola corretta. L'elenco di candidati viene poi mescolato, garantendo che la posizione della parola corretta sia casuale.

3. *Preprocessing del dataset*: la funzione `preprocess_function` prepara il dataset per il task di Multiple Choice. In particolare, il contesto (cioè la frase troncata senza l'ultima parola) viene ripetuto per ciascuna delle quattro opzioni di completamento, creando coppie di tipo domanda-opzione che saranno successivamente tokenizzate. Ogni coppia è composta dalla frase senza l'ultima parola e una delle quattro opzioni come potenziale completamento.

4. *Tokenizzazione*: utilizzando il tokenizer pre-addestrato, ogni coppia domanda-opzione viene tokenizzata con padding e truncation, assicurando che tutte le sequenze siano della stessa lunghezza. Le frasi e le opzioni vengono combinate in un formato tokenizzato adeguato per il task.

5. *Riorganizzazione dei dati per il task di Multiple Choice*: i dati tokenizzati vengono strutturati in batch di quattro opzioni per ciascuna frase, in modo da poter utilizzare l'architettura del modello per valutare ciascuna opzione in parallelo.

6. *Applicazione del pre-processing al dataset*: la funzione di pre-processing viene applicata in batch al dataset filtrato e mischiato.

3.4.4 Fine tuning

Il fine-tuning è una fase fondamentale per adattare un modello pre-addestrato al task specifico di generazione e comprensione del dialetto fermano.

Per ciascuno dei task sono state adottate tecniche di fine-tuning specifiche, in modo da sfruttare al meglio le capacità dei due modelli distinti.

Per quanta riguarda Minerva, l'implementazione del fine tuning è riportata nel Listato 3.6.

Il modello pre-addestrato è stato caricato utilizzando *AutoModelForCausalLM*, una classe di Hugging Face specifica per il language modeling causale. Minerva è un esempio di tali causal language model in quanto prevede il token successivo in una sequenza di parole in modo regressivo, cioè guardando solo i token precedenti.

Per preparare i dati di input, è stata utilizzata la classe *DataCollatorForLanguageModeling* di Hugging Face. Un data collator prende in input una lista di esempi (ad esempio, frasi) e li raggruppa in batch, applicando eventuali operazioni di pre-processing necessarie. In particolare, il data collator si occupa di operazioni come il padding, che uniforma le sequenze di input alla stessa lunghezza, o il masking, che maschera casualmente alcuni token per l'addestramento nei task di language modeling. L'obiettivo principale del data collator è preparare i dati in un formato che sia compatibile con il modello, ottimizzando la gestione degli input nei batch e facilitando il processo di training. Nel caso del modello Minerva, il parametro `mlm` è impostato a False, poiché nel causal language modeling non è richiesto il masking delle parole (come avviene in BERT).

Sono stati, inoltre, configurati i parametri per il training, come il learning rate, il batch size e il numero di epoche.

Inoltre, il dataset è stato suddiviso in set di training (85%), validation (10%) e test (5%), consentendo di monitorare le performance del modello e di verificare la sua capacità di generalizzare su dati non visti.

Il training è stato gestito con la classe *Trainer* di Hugging Face, che permette di monitorare l'andamento dell'addestramento e di gestire automaticamente il salvataggio dei checkpoint. Al termine del processo, il modello fine-tuned e il tokenizer sono stati salvati per le fasi successive, come la valutazione.

```
# Carica il modello pre-addestrato
model = AutoModelForCausalLM.from_pretrained("sapienzanlp/Minerva-350M-base-v1.0")

# Imposta il Data Collator per il language modeling
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

# Definisce i parametri di training
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=2,
    weight_decay=0.01,
    report_to="none"
)

# Suddivide il dataset in 85% training e 15% restante (da dividere in validation e test)
train_val_test_split = tokenized_datasets['train'].train_test_split(test_size=0.15)
train_dataset = train_val_test_split['train']  # 85% training
val_test_dataset = train_val_test_split['test']  # 15% restante

# Divide ulteriormente il 15% restante in 10% validation e 5% test (usato per TWP e TWS)
val_test_split = val_test_dataset.train_test_split(test_size=0.33)
validation_dataset = val_test_split['train']  # 10% validation (2/3 del 15%)
test_dataset = val_test_split['test']  # 5% test (1/3 del 15%)
print(f"Dimensione del test set: {len(test_dataset)}")

# Inizializza il Trainer per l'addestramento del modello
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=validation_dataset,
    data_collator=data_collator,
)

# Avvia l'addestramento
trainer.train()
```

Listato 3.6: Fine tuning di Minerva

Per il task di Target Word Prediction, BERT è stato caricato utilizzando la classe *AutoModelForMaskedLM* di Hugging Face specifica per i masked language model che prevedono un token mascherato in una sequenza in relazione al contesto dell'intero input.

Il Data Collator è analogo a quello utilizzato per il modello Minerva, con la differenza che il parametro `mlm` è impostato a True ed è stata specificata una probabilità di mascheramento di 0.15. Questo significa che il 15% delle parole nella frase è stato mascherato, permettendo al modello di imparare a predire il token mancante.

Per quanto riguarda il task di Target Word Selection, Bert è stato caricato utilizzando la classe *AutoModelForMultipleChoice* di Hugging Face specifica per i modelli Multiple Choice, in cui il modello valuta una serie di risposte candidate per selezionare quella più adatta in base al contesto fornito.

Infine, poiché Hugging Face non offre un Data Collator specifico per i task di scelta multipla, è stato necessario effettuare un adattamento. Seguendo le linee guida di Hugging Face, è stato utilizzato e adattato il *DataCollatorWithPadding* per soddisfare le esigenze di input dei modelli di scelta multipla. Tale Data Collator è progettato per gestire il padding delle sequenze in modo che tutte abbiano la stessa lunghezza all'interno di un batch. Tuttavia, poiché per il task di scelta multipla ogni input deve essere composto da una domanda e da una serie di opzioni candidate, il collator è stato personalizzato per strutturare i dati in modo

che ogni batch contenga gruppi di domande e opzioni, con il padding applicato su ciascuna opzione. L'adattamento consiste nell'organizzare ogni coppia "domanda-opzione" in modo che il modello possa valutarle separatamente e assegnare una probabilità a ciascuna opzione. Questo tipo di pre-processing permette al modello di trattare ogni opzione come un input indipendente, facilitando la scelta dell'opzione corretta.

3.5 Valutazione

La valutazione delle performance dei modelli è stata suddivisa in tre diverse analisi: una valutazione qualitativa per la generazione di testo, una valutazione quantitativa per il task di Target Word Prediction (TWP) e una per il Target Word Selection (TWS). Queste due ultime valutazioni sono state eseguite su tre dataset distinti:

- test set in dialetto, al fine di analizzare la capacità del modello di generalizzare a nuovi dati nel dialetto specifico;
- train set in dialetto, per valutare come il modello si comporta sui dati con cui è stato addestrato, testando se mostra coerenza, accuratezza e capacità di apprendere;
- dataset italiano (*"silvia-casola/WITS"*) utile per verificare se il modello pre-addestrato perde le sue conoscenze pregresse.

Questo confronto su vari dataset fornisce un quadro più completo della capacità del modello di adattarsi e mostrare performance elevate su dialetti specifici, mantenendo, al contempo, una buona comprensione del linguaggio standard.

3.5.1 Valutazione qualitativa

Per il task di generazione di testo, non si dispone di metriche oggettive per valutare la qualità dell'output. Il testo generato dal modello può essere valutato da parlanti nativi del dialetto fermano, utilizzando una scala da 1 a 5 per misurare due aspetti fondamentali: adeguatezza e fluidità. L'adeguatezza valuta quanto il testo generato è coerente e risponde in modo corretto all'input dato mentre la fluidità misura quanto il testo generato è grammaticalmente corretto e naturale.

La procedura implementata è mostrata nel Listato 3.7.

```
# Inizializza la pipeline di generazione di testo
pipeline = transformers.pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    device=0,
)

# Testa il modello con un input di esempio
input_text = "La capitale dell'Italia è"

# Pipeline di generazione di testo con Beam Search
output_beam_search = pipeline(
  input_text,
  max_new_tokens=128,
  num_beams=5,  # numero di beam, maggiore è il numero più opzioni vengono esplorate
  early_stopping=True  # ferma la ricerca se tutte le beam convergono alla stessa sequenza
)

# Stampa il risultato con Beam Search
print("Beam Search Output:")
print(output_beam_search[0]['generated_text'])

# Pipeline di generazione di testo con sampling
output = pipeline(
  input_text,
  max_new_tokens=128,
  do_sample=True,  # abilita il sampling
  temperature=0.7,  # aumenta la diversità del testo; un valore più alto genera output più creativi
  top_k=50,  # limita la scelta del modello ai top 50 token più probabili
```

```
    top_p=0.9,  # usa il nucleus sampling, considerando solo i token la cui probabilità cumulativa raggiunge il 90%
    repetition_penalty=1.2,  # penalizza le ripetizioni
)

# Stampa il risultato con sampling
print("Sampling Output:")
print(output[0]['generated_text'])
```

Listato 3.7: Valutazione qualitativa

Dopo aver caricato il modello e il tokenizer fine-tuned, è stata inizializzata una pipeline di generazione di testo.

Per testare la capacità di generazione del modello, è stato fornito un esempio di input generico, come "La capitale dell'Italia è". In questo contesto, sono stati confrontati due metodi di decoding: Beam Search e i metodi di Sampling (Top-k e Top-p).

La Beam Search è una tecnica che esplora diversi cammini per generare la sequenza più probabile; l'early stopping è stato abilitato per terminare la generazione una volta che tutte le beam convergono alla stessa sequenza. La Beam Search è particolarmente utile per ottenere risultati coerenti e grammaticalmente corretti, ma potrebbe risultare meno creativa.

Il Sampling viene utilizzato per produrre output più vari e creativi. In particolare, il Top-k limita la scelta ai primi k token più probabili, mentre il Top-p considera i token fino a quando la probabilità cumulativa raggiunge una soglia predeterminata (ad esempio, il 90%). Il parametro `temperature` aumenta la diversità delle parole generate mentre `repetition_penalty` è utile per ridurre le ripetizioni e migliorare la fluidità del testo.

3.5.2 Valutazione TWP

La valutazione del task di Target Word Prediction (TWP) è stata effettuata tramite un'analisi quantitativa per misurare la capacità del modello di predire correttamente l'ultima parola di una frase in dialetto fermano.

L'implementazione della valutazione del TWP per Minerva sul test set in dialetto è riportata nel Listato 3.8.

Innanzitutto è stata troncata l'ultima parola di ciascuna frase del test set. La parola troncata è considerata la parola corretta e viene confrontata con la parola predetta dal modello. Nel caso di Bert, oltre a rimuovere l'ultima parola, è stato inserito il token [MASK] per mascherare l'ultima parola. Questo approccio permette a BERT di predire la parola mancante in modo efficace.

Per ogni frase troncata, il modello fine-tuned genera le top-5 parole più probabili per la posizione successiva. Le parole predette sono state ordinate per probabilità e la più probabile è stata selezionata come la parola predetta principale. Inoltre, sono stati verificati i top-3 e top-5 risultati per determinare se la parola corretta si trovasse tra le prime 3 o 5 predizioni.

Per Minerva la selezione delle top-5 parole più probabili avviene analizzando i logit generati in output. Durante la predizione, il modello produce un insieme di valori numerici (i logit) per tutti i possibili token del vocabolario, valutandoli come possibili candidati per la posizione successiva nella sequenza di testo. I logit rappresentano dei punteggi non normalizzati, che corrispondono alla tendenza del modello a predire ciascun token; maggiore è il logit, più alta è la preferenza del modello per quel token nella sequenza attuale. I logit vengono ordinati per identificare i punteggi più alti. A questo punto, vengono estratti i top-5 logit che corrispondono ai token più probabili. Gli ID dei token con i punteggi più alti vengono, quindi, convertiti nelle rispettive parole (o token di testo) usando il tokenizer. Questi token sono ordinati dalla parola più probabile (con il logit maggiore) alla meno probabile all'interno delle prime cinque predizioni.

Con BERT, invece, piuttosto che i logit, vengono utilizzati gli score generati quando viene predetta la parola mascherata.

Le metriche di valutazione calcolate sono state l'accuratezza complessiva, Top-3 e Top-5 accuracy, e la similarità del coseno media. L'accuratezza complessiva mostra la precisione con cui il modello riesce a predire esattamente la parola target. Essa è la percentuale di corrispondenze esatte tra le parole predette e quelle corrette, normalizzando entrambe le parole per rimuovere la punteggiatura e ignorare le differenze di maiuscole o minuscole. La formula utilizzata è:

$$\text{Accuracy} = \frac{\text{Numero di predizioni corrette}}{\text{Numero totale di predizioni}}$$

La top-3 accuracy e la top-5 accuracy sono state calcolate contando i casi in cui la parola corretta appariva tra le prime 3 e 5 predizioni, rispettivamente.

Per valutare la similarità semantica tra la parola predetta e quella corretta, è stato calcolato il valore medio della similarità del coseno tramite il modello *Sentence-BERT*. Questo punteggio quantifica quanto la parola predetta sia vicina al significato della parola corretta, anche quando non è esattamente la stessa. La similarità del coseno è stata calcolata tra i vettori embedding della parola predetta e della parola corretta nel seguente modo:

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

dove:

- $\vec{a} \cdot \vec{b}$ rappresenta il prodotto scalare dei vettori;
- $\|\vec{a}\|$ e $\|\vec{b}\|$ sono le norme (moduli) dei vettori $\vec{a}$ e $\vec{b}$.

I vettori $\vec{a}$ e $\vec{b}$ sono gli embedding delle parole predetta e corretta e tale misura varia tra -1 e 1 con valori vicini a 1 che indicano forte similarità.

```python
# Inizializza Sentence-BERT per la similarità coseno
sbert_model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

# Utilizza il test_dataset per TWP
test_sentences = test_dataset['text']  # Lista di frasi dal test set

# Liste per le predizioni e le parole corrette
predicted_words = []
correct_words = []
similarities = []
top3_correct = 0  # Conta le predizioni corrette nella top-3
top5_correct = 0  # Conta le predizioni corrette nella top-5

# Funzione per troncare la frase prima della parola target
def truncate_last_word(conversation):
    words = conversation.split()
    truncated_sentence = ' '.join(words[:-1])  # Tronca l'ultima parola
    return truncated_sentence, words[-1]

# Funzione per verificare se la parola è valida (min 3 caratteri e non punteggiatura)
def is_valid_word(word):
    word = re.sub(r'[^\w\s]', '', word)  # Rimuove punteggiatura
    return len(word) >= 3

# Funzione per rimuovere punteggiatura e normalizzare il testo (minuscolo)
def normalize_word(word):
    word = word.lower()  # Converti in minuscolo
    word = re.sub(r'[^\w\s]', '', word)  # Rimuove punteggiatura
    word = word.strip()  # Rimuove spazi extra
    return word

# Funzione per calcolare la similarità coseno
def calculate_similarity(predicted_word, correct_word):
    predicted_embedding = sbert_model.encode([predicted_word])
    correct_embedding = sbert_model.encode([correct_word])
    similarity = cosine_similarity(predicted_embedding, correct_embedding)
    return similarity[0][0]

# Valutazione su tutte le frasi non viste
for sentence in test_sentences:
    truncated_sentence, correct_word = truncate_last_word(sentence)

    # Verifica se la parola corretta è valida (almeno 3 caratteri e non punteggiatura)
    if not is_valid_word(correct_word):
        print(f"Frase scartata: {sentence} (parola da predire '{correct_word}' non valida)")
        continue  # Salta questa frase
```

```
47
48      print(f"Frase troncata: {truncated_sentence}")
49
50      # Tokenizza la frase troncata
51      inputs = tokenizer(truncated_sentence, return_tensors="pt").to(model.device)
52
53      # Predice i top-5 token più probabili
54      with torch.no_grad():
55          outputs = model(**inputs)
56          logits = outputs.logits[0, -1, :]  # Ottiene i logits per il token successivo
57          top_k_values, top_k_indices = torch.topk(logits, k=5)  # Ottiene i top-5 token
58
59      # Converti i token ID in parole
60      predicted_words_list = [tokenizer.decode([idx]).strip() for idx in top_k_indices]
61
62      # Memorizza la prima parola (la più probabile)
63      predicted_word = predicted_words_list[0]
64      predicted_words.append(predicted_word)
65      correct_words.append(correct_word)
66
67      # Verifica se la parola corretta è nelle top-3 e top-5 predizioni
68      if normalize_word(correct_word) in [normalize_word(w) for w in predicted_words_list[:3]]:
69          top3_correct += 1
70      if normalize_word(correct_word) in [normalize_word(w) for w in predicted_words_list[:5]]:
71          top5_correct += 1
72
73      # Calcola la similarità tra la parola predetta e quella corretta
74      similarity = calculate_similarity(predicted_word, correct_word)
75      similarities.append(similarity)
76
77      print(f"Parola predetta: {predicted_word}")
78      print(f"Parola corretta: {correct_word}")
79      print(f"Similarità Coseno: {similarity:.2f}")
80      print("---")
81
82  # Funzione aggiornata per calcolare l'accuracy complessiva
83  def calculate_accuracy(predicted_words, correct_words):
84      correct_count = sum([1 for pred, correct in zip(predicted_words, correct_words)
85                           if normalize_word(pred) == normalize_word(correct)])
86      accuracy = correct_count / len(correct_words)
87      return accuracy
88
89  # Calcolo dell'accuracy complessiva
90  accuracy = calculate_accuracy(predicted_words, correct_words)
91  print(f"Accuracy complessiva: {accuracy:.2f}")
92
93  # Calcolo della top-3 e top-5 accuracy
94  top3_accuracy = top3_correct / len(correct_words)
95  top5_accuracy = top5_correct / len(correct_words)
96
97  print(f"Top-3 Accuracy: {top3_accuracy:.2f}")
98  print(f"Top-5 Accuracy: {top5_accuracy:.2f}")
99
100 # Calcolo della similarità media
101 average_similarity = sum(similarities) / len(similarities)
102 print(f"Similarità media: {average_similarity:.2f}")
```

Listato 3.8: Valutazione della TWP

3.5.3 Valutazione TWS

Nel task di Target Word Selection (TWS), il modello è chiamato a selezionare la parola corretta tra un insieme di opzioni, in base al contesto fornito. Per la valutazione del modello, sono state utilizzate le medesime metriche del TWP, cioè accuratezza complessiva, Top-3 e Top-5 accuracy e similarità del coseno media. La differenza sta nel modo in cui il modello fine-tuned genera la predizione, ovvero, in questo caso, esso seleziona la parola corretta tra le varie candidate. Il codice che si occupa di eseguire ciò è riportato nel Listato 3.9.

Il codice inizia iterando attraverso ciascun esempio nel test set, dove ogni esempio contiene la frase da valutare (`input_ids`, `attention_mask`) e la parola corretta tra le opzioni.

Per ogni esempio, vengono estratti `input_ids` e `attention_mask` e questi vengono raggruppati in un dizionario degli input. Questi rappresentano l'input tokenizzato e il mascheramento dei token, che indica al modello quali token considerare significativi.

Il modello viene eseguito in modalità di valutazione (`torch.no_grad()`), che disabilita il calcolo dei gradienti per ridurre l'uso di memoria e velocizzare l'inferenza.

Passando input e label al modello, vengono generati i logit come output. La classe predetta (ovvero l'opzione più probabile) è quella con il logit più alto. L'indice `predicted_class`

è utilizzato per selezionare la parola predetta (`predicted_word`) dall'array delle opzioni presenti in `example['options']`.

```
predicted_words = []
correct_words = []
similarities = []

top3_correct = 0
top5_correct = 0

for i, example in enumerate(test_dataset):
    inputs = {
        'input_ids': torch.tensor(example['input_ids']),
        'attention_mask': torch.tensor(example['attention_mask'])
    }
    labels = torch.tensor([example['labels']])

    with torch.no_grad():
        outputs = model(**{k: v.unsqueeze(0) for k, v in inputs.items()}, labels=labels)
        logits = outputs.logits

    predicted_class = logits.argmax().item()
    predicted_word = example['options'][predicted_class]

    predicted_words.append(predicted_word)
    correct_words.append(example['correct_word'])

    similarity = calculate_similarity(predicted_word, example['correct_word'])
    similarities.append(similarity)

    # Calcola top-3 e top-5 accuracy
    top3_correct += top_k_accuracy(logits, labels, k=3)
    top5_correct += top_k_accuracy(logits, labels, k=5)

# Calcola e stampa i risultati
accuracy = calculate_accuracy(predicted_words, correct_words)
average_similarity = sum(similarities) / len(similarities)
top3_accuracy = top3_correct / len(test_dataset)
top5_accuracy = top5_correct / len(test_dataset)
```

Listato 3.9: Valutazione della TWS

CAPITOLO 4

Test e valutazione dei risultati

Il capitolo sarà dedicato all'analisi dei risultati ottenuti dai modelli Minerva e BERT applicati ai task di Target Word Prediction (TWP) e Target Word Selection (TWS). L'obiettivo principale di questa fase sarà valutare le capacità dei modelli di comprendere e generare contenuti linguistici in contesti specifici, confrontando le performance pre e post fine-tuning su dataset in dialetto fermano e in lingua italiana. Verranno, quindi, mostrati e discussi i risultati qualitativi e quantitativi in relazione a diverse epoche di addestramento al fine di indagare sul bilanciamento tra generalizzazione e overfitting dei modelli. L'analisi fornirà una panoramica completa delle potenzialità e dei limiti dei modelli, mostrando i progressi raggiunti attraverso le tecniche di fine-tuning adottate e i limiti ancora sfidanti.

4.1 Risultati TWP con Minerva

La prima valutazione del modello è stata eseguita su Minerva prima del fine-tuning, per ottenere una baseline di riferimento e identificare le aree di miglioramento. Questa fase iniziale è fondamentale poiché permette di confrontare le prestazioni del modello pre e post fine-tuning e di valutare l'efficacia del processo di ottimizzazione. I risultati ottenuti, mostrati in Tabella 4.1, evidenziano le limitazioni del modello Minerva pre-addestrato quando viene applicato a dati dialettali e, in misura minore, a testi in italiano standard.

metriche	**dataset dialetto**	**dataset italiano**
Accuracy complessiva	0.02	0.20
Top-3 Accuracy	0.03	0.26
Top-5 Accuracy	0.04	0.30
Similarità media	0.28	0.42

Tabella 4.1: Risultati della valutazione del TWP per Minerva prima del fine tuning

Come si può osservare, le metriche sui dati in dialetto risultano quasi nulle. L'accuracy complessiva è solo del 2%, e metriche aggiuntive, come la Top-3 e la Top-5 Accuracy, così come la similarità del coseno media, indicano chiaramente che Minerva, in questa configurazione, non è in grado di fare previsioni adeguate nel dialetto specifico. Questo risultato è prevedibile, poiché il modello non è stato addestrato su questo tipo di dati, e quindi non ha sviluppato la capacità di comprendere e predire efficacemente termini dialettali o le particolarità linguistiche del dialetto marchigiano.

Sul dataset italiano, i risultati sono leggermente migliori. L'accuracy complessiva sale al 20%, la Top-3 e la Top-5 Accuracy raggiungono, rispettivamente, valori di 26% e 30%, mentre la similarità del coseno media è 0.42. Tuttavia, questi valori restano ben al di sotto di quanto ci si aspetterebbe da un modello ottimizzato, confermando che il modello Minerva, anche in presenza di un dataset italiano, non presenta prestazioni elevate.

Questi risultati indicano che Minerva, nella configurazione iniziale, non eccelle nemmeno sui dati italiani standard. Di conseguenza, non ci si può attendere un miglioramento significativo delle sue capacità predittive esclusivamente attraverso il fine-tuning. Infatti, il modello non possiede una base sufficientemente solida per gestire task complessi o linguaggi sottorappresentati senza un processo di ottimizzazione molto specifico e mirato.

Tuttavia, ciò non sminuisce l'importanza del fine-tuning, ma al contrario sottolinea la sua necessità per adattare il modello alle peculiarità linguistiche e culturali del dialetto fermano. L'obiettivo del fine-tuning, dunque, non è solo migliorare le metriche, ma anche verificare quanto il modello possa apprendere, ed eventualmente adattarsi, a un dominio nuovo partendo da basi limitate.

Dopo aver eseguito il fine-tuning come descritto nel capitolo precedente, è stata condotta una valutazione qualitativa del testo generato da Minerva per analizzare l'impatto del fine-tuning sulle sue capacità di produzione linguistica. Questo tipo di valutazione permette di osservare e analizzare come il modello risponde a un input dato e in che misura riesce a generare testo, utilizzando due metriche: adeguatezza e fluidità. L'adeguatezza valuta la capacità del modello di rispondere in modo coerente e pertinente all'input, mentre la fluidità si riferisce alla correttezza grammaticale e alla naturalezza del linguaggio prodotto. Ogni generazione è valutata su una scala da 1 a 5 per ciascuna delle due metriche.

Con la tecnica del Beam Search, con input "La capitale dell'Italia è", un esempio di frase prodotta dal modello fine-tuned addestrato per 2 epoche è:

«*La capitale dell'Italia è la città de Fermo. E' la città più bella d'Italia! E' la città più bella d'Italia! E' la città più bella d'Italia! ...*»

Il testo generato utilizzando la tecnica del Beam Search, sia per il fine-tuning su 2 epoche che per le successive, ha mostrato una marcata tendenza alla ripetitività, con frasi che si reiterano senza aggiungere contenuto significativo. Per questo motivo, la valutazione qualitativa si è concentrata principalmente sugli output prodotti tramite la tecnica del Sampling, che garantisce una maggiore varietà e creatività nella generazione del testo.

L'input scelto per il test, ovvero "La capitale dell'Italia è", è stato pensato strategicamente per valutare due aspetti fondamentali: la capacità del modello di preservare le conoscenze acquisite durante il pre-addestramento e la sua abilità di adattarsi alle nuove informazioni introdotte attraverso il fine-tuning. Questa frase semplice e generica consente di osservare se il modello riesce a mantenere informazioni linguistiche e culturali di base, come quelle apprese dai dataset originali, e al contempo permette di verificare in che misura il fine-tuning influisce sulla generazione di risposte che riflettano il dialetto fermano o i dati specifici con cui è stato addestrato.

I risultati qualitativi delle generazioni sono stati sintetizzati nella Tabella 4.2.

epoche	adeguatezza (1-5)	fluidità (1-5)	commento
2	1	1	Frasi incoerenti e sconnesse
3	1.5	1	Leggero miglioramento della coerenza
4	1.5	1.5	Uso più plausibile del dialetto
5	2.5	2	Risultati più coerenti, ma ancora limitati
6	2	2	Non rilevati miglioramenti

Tabella 4.2: Valutazione qualitativa della generazione di testo per diverse epoche di addestramento

Dopo 2 epoche, l'output generato presenta evidenti problemi di coerenza e fluidità. Ad esempio, con input «La capitale dell'Italia è», il modello risponde con: «*La capitale dell'Italia è londana, e se non ve lo dico io me fai la cura te senti...*». Questo risultato riflette una mancanza di comprensione contestuale: il modello non completa la frase con un'informazione logica e pertinente, generando invece frasi sconnesse e prive di significato, giustificando il punteggio minimo di 1 sia per adeguatezza che per fluidità.

Con il fine-tuning su 3 epoche, si osservano miglioramenti nella capacità del modello di generare risposte più coerenti. Un esempio significativo è: «*La capitale dell'Italia è l'Umbria. Essa, a dijolo, è la côla vòna de lu Paese!*». Sebbene l'output non sia ancora del tutto corretto, si nota che il modello tenta di completare la frase con una località geografica, dimostrando una comprensione basilare dell'input. Questo giustifica un incremento nei punteggi di adeguatezza a 1.5, pur mantenendo un valore basso di 1 per fluidità a causa della sintassi ancora poco naturale.

Con 4 epoche, nonostante i miglioramenti nel tentativo di generare una narrazione più strutturata e l'uso più plausibile del dialetto, il contenuto rimane confuso e privo di una chiara coerenza rispetto all'input. Tuttavia, si può notare un leggero progresso nella fluidità linguistica in quanto il modello cerca di concatenare frasi connesse come: «*Penzó sulo a tanta jente se cerca de staje lontano ... Ah sci? Sci! Perché 'n cià da ji su 'na città cuscì grossa*».

A partire dalla quinta epoca, si osservano risultati più convincenti. Ad esempio, l'output «*La capitale dell'Italia è la città de l'amore. Ma a Roma, per amor di'stu segretu lu tinia veramente...*» mostra un significativo miglioramento nella coerenza. Il modello identifica correttamente che l'input si riferisce a Roma e fornisce una risposta più contestualizzata e fluida, giustificando punteggi più elevati di 2.5 per adeguatezza e 2 per fluidità.

Con 6 epoche di addestramento, il modello mostra segni di regressione rispetto agli avanzamenti visti con 5 epoche. Sebbene la struttura grammaticale rimanga consistente, l'output generato, come «*La capitale dell'Italia è grande e grande sta dentro la casa d'un contadi...*», suggerisce che il modello inizia a perdere parte delle conoscenze pregresse acquisite durante il pre-addestramento. La coerenza diminuisce significativamente, con risposte che si allontanano dall'input fornito e che risultano meno pertinenti rispetto a quelle prodotte nelle epoche precedenti. Questo decremento è attribuibile all'effetto noto come catastrophic forgetting, in cui il modello sovrascrive informazioni precedentemente apprese durante il fine-tuning su un dataset specifico. Dal punto di vista della fluidità, i risultati rimangono stazionari rispetto alle epoche precedenti, senza mostrare ulteriori miglioramenti. Ciò indica che, sebbene il modello sia riuscito a stabilizzare alcune capacità linguistiche, il fine-tuning non riesce a raffinare ulteriormente la naturalezza e la scorrevolezza del testo generato. Pertanto, i punteggi di 2 sia per adeguatezza che per fluidità riflettono queste osservazioni, evidenziando uno stallo nella qualità complessiva del testo prodotto.

In generale, nonostante l'aumento del numero di epoche di addestramento, il testo generato da Minerva ha continuato a mostrare limitazioni significative. In particolare, il modello spesso produce testo non pertinente, incoerente o completamente scorretto rispetto all'input iniziale.

Questi risultati suggeriscono che, sebbene la fluidità migliori leggermente con l'aumento delle epoche, il testo rimane artificioso e poco naturale, con evidenti errori grammaticali. Pur rappresentando un piccolo progresso, l'approccio attuale non è sufficiente per ottenere un modello robusto e adatto alla generazione in dialetto fermano.

I risultati ottenuti dalla valutazione quantitativa del task di Target Word Prediction per un numero variabile di epoche di fine-tuning sono riportati nella Tabella 4.3.

I risultati sul test set in dialetto mostrano un'accuracy complessiva bassa, con valori che si aggirano intorno a 0.07 nelle prime due epoche e che diminuiscono leggermente con l'aumentare delle epoche di addestramento.

dataset	metriche	2 epoche	3 epoche	4 epoche	5 epoche	6 epoche
Test Dialetto	Accuracy Complessiva	0.07	0.07	0.05	0.05	0.05
	Top-3 Accuracy	0.10	0.11	0.08	0.08	0.08
	Top-5 Accuracy	0.12	0.13	0.10	0.09	0.09
	Similarità Media	0.30	0.30	0.29	0.28	0.29
Train Dialetto	Accuracy Complessiva	0.15	0.23	0.29	0.31	0.32
	Top-3 Accuracy	0.19	0.27	0.32	0.33	0.34
	Top-5 Accuracy	0.21	0.29	0.32	0.33	0.34
	Similarità Media	0.37	0.46	0.53	0.59	0.59
Italiano	Accuracy Complessiva	0.16	0.13	0.12	0.11	0.11
	Top-3 Accuracy	0.21	0.20	0.18	0.16	0.17
	Top-5 Accuracy	0.23	0.22	0.20	0.18	0.19
	Similarità Media	0.38	0.36	0.35	0.34	0.34

Tabella 4.3: Risultati della valutazione del TWP per Minerva su diverse epoche

Anche le metriche di top-3 e top-5 accuracy riflettono una performance limitata, con massimi, rispettivamente, di 0.11 e 0.13. La similarità media si mantiene stabile intorno a 0.30, indicando che il modello riesce a generare parole moderatamente vicine a quella corretta secondo la similarità del coseno. Tuttavia, il basso livello di accuracy suggerisce che il modello fatica a generalizzare sul dialetto.

Al contrario, i risultati sul train set in dialetto evidenziano un netto miglioramento con l'aumentare delle epoche. L'accuracy complessiva cresce da 0.15 a 0.32, indicando che il modello sta apprendendo dai dati di training. Anche le metriche top-3 e top-5 accuracy seguono un andamento simile, con valori, rispettivamente, di 0.19 e 0.21 alla prima epoca e 0.34 alla sesta epoca. La similarità media mostra un progresso significativo, passando da 0.37 nelle prime epoche a 0.59 alla fine, evidenziando che il modello riesce a generare parole sempre più simili a quelle corrette nel contesto del training.

Sul dataset italiano, invece, si osserva un andamento opposto: tutte le metriche diminuiscono progressivamente con l'aumentare delle epoche. L'accuracy complessiva decresce da 0.16 a 0.11, e la similarità media scende da 0.38 a 0.34. Questi dati suggeriscono che il fine-tuning sta compromettendo le conoscenze pregresse del modello, portando a una perdita di generalizzazione per il linguaggio italiano standard. Questo comportamento evidenzia un problema di overfitting, dove il modello si adatta eccessivamente ai dati di training in dialetto, penalizzando la capacità di generalizzare su nuovi dati.

Un esempio in cui il modello è riuscito a predire correttamente la parola target è il seguente:

- *Frase troncata:* «Bisogna porta' la testa su le»
- *Parola predetta:* «spalle»
- *Parola corretta:* «spalle»

In questo caso, il modello ha dimostrato una comprensione contestuale precisa, fornendo una predizione coerente con il significato della frase. Questo risultato mostra la capacità del modello di apprendere alcune dipendenze sintattiche e semantiche dal dataset.

Viceversa, un esempio di predizione errata è il seguente:

- *Frase troncata:* «U sciarbu è quillu che fa i»
- *Parola predetta:* «fatti»
- *Parola corretta:* «muratu'»

Qui, il modello ha fallito nella comprensione del contesto, generando una parola semanticamente distante dalla parola corretta. Questo suggerisce che il modello potrebbe avere difficoltà con termini specifici del dialetto o con frasi meno frequenti nel dataset di training.

In alcuni casi, il modello ha generato parole diverse da quella corretta, ma comunque semanticamente o concettualmente accettabili. Ad esempio:

- *Frase troncata:* «Purtimo 'sta robba dentro 'lla»
- *Parola predetta:* «casa»
- *Parola corretta:* «capanna»

In questo caso, il modello ha scelto una parola che, pur non essendo identica a quella corretta, mantiene una certa coerenza con il contesto. Questo tipo di errore indica che il modello riesce a catturare il significato generale della frase, anche se non fornisce la parola esatta.

4.2 Risultati TWP con BERT

Analogamente a quanto eseguito per Minerva, per comprendere le prestazioni iniziali di BERT nel task di TWP, è stata effettuata una valutazione del modello nella sua versione pre-addestrata. I risultati sono riportati nella Tabella 4.4.

metriche	**dataset dialetto**	**dataset italiano**
Accuracy complessiva	0.03	0.31
Top-3 Accuracy	0.06	0.39
Top-5 Accuracy	0.07	0.42
Similarità media	0.28	0.50

Tabella 4.4: Risultati della valutazione del TWP per BERT prima del fine tuning

Le metriche sui dati in dialetto rivelano prestazioni quasi nulle come osservato anche per Minerva. Con un'accuracy complessiva del 3% e valori leggermente superiori nelle metriche Top-3 e Top-5, il modello evidenzia, ovviamente, difficoltà significative nell'affrontare il linguaggio dialettale.

Sul dataset italiano, il modello mostra risultati migliori con un'accuracy complessiva del 31% e una similarità media del 50%, ad indicare una comprensione di base della lingua italiana. Tuttavia, anche in questo caso, le performance non sono ottimali suggerendo che il fine-tuning per il dialetto fermano potrebbe non raggiungere performance elevate senza significativi interventi sul dataset o sull'architettura.

Sebbene BERT (così come Minerva) sia stato pre-addestrato su grandi quantità di dati in italiano, come articoli di Wikipedia e Common Crawl, i risultati ottenuti sul dataset italiano indicano prestazioni inferiori alle aspettative. Questo fenomeno può essere attribuito a diversi fattori. In primo luogo, il dataset italiano utilizzato per il test consiste in riassunti di articoli di Wikipedia, che presentano una struttura sintetica e densa di informazioni, differente dal testo completo su cui BERT è stato addestrato. Questa discrepanza potrebbe influire sulla capacità del modello di comprendere e completare efficacemente il testo.

Inoltre, BERT eccelle nella previsione di parole centrali, sfruttando il contesto bidirezionale (parole precedenti e successive). Tuttavia, nel task di TWP, la parola da predire si trova alla fine della frase, privando il modello del contesto a destra. Questa configurazione potrebbe non essere ottimale rispetto ai suoi meccanismi di pre-addestramento.

La Tabella 4.5 riporta i risultati ottenuti per il task di Target Word Prediction utilizzando BERT per il fine-tuning per diverse epoche.

dataset	metriche	2 epoche	3 epoche	4 epoche	5 epoche	6 epoche
Test Dialetto	Accuracy Complessiva	0.05	0.06	0.08	0.08	0.10
	Top-3 Accuracy	0.10	0.10	0.13	0.13	0.17
	Top-5 Accuracy	0.13	0.14	0.16	0.16	0.19
	Similarità Media	0.28	0.28	0.29	0.29	0.31
Train Dialetto	Accuracy Complessiva	0.07	0.09	0.09	0.10	0.10
	Top-3 Accuracy	0.12	0.14	0.14	0.16	0.16
	Top-5 Accuracy	0.14	0.17	0.17	0.19	0.20
	Similarità Media	0.29	0.30	0.30	0.31	0.31
Italiano	Accuracy Complessiva	0.26	0.28	0.28	0.27	0.27
	Top-3 Accuracy	0.35	0.36	0.36	0.36	0.36
	Top-5 Accuracy	0.38	0.39	0.39	0.40	0.40
	Similarità Media	0.46	0.48	0.48	0.48	0.47

Tabella 4.5: Risultati della valutazione del TWP per BERT su diverse epoche

Il modello mostra un miglioramento graduale nel tempo, con un'accuracy complessiva che passa da 0.05 (2 epoche) a 0.10 (6 epoche). Tuttavia, le metriche rimangono basse in termini assoluti, indicando che il modello fatica ad apprendere efficacemente le peculiarità del dialetto. Anche le metriche Top-3 e Top-5 accuracy mostrano un incremento simile, con valori massimi, rispettivamente, di 0.17 e 0.19 dopo 6 epoche. Questo suggerisce che, pur migliorando, il modello non riesce a generalizzare pienamente al dialetto.

Tale considerazione trova ulteriore conferma nei risultati ottenuti sul training set. Infatti, le metriche relative a BERT mostrano un incremento decisamente meno marcato rispetto a quanto osservato per Minerva, dove l'accuracy complessiva raddoppia tra la seconda e la sesta epoca di addestramento. Questo comportamento suggerisce che BERT incontra maggiori difficoltà nell'apprendere efficacemente le caratteristiche distintive del dialetto fermano.

A differenza del dataset dialettale, i risultati ottenuti sul dataset italiano evidenziano valori più elevati e una maggiore stabilità nel corso delle epoche di addestramento. In particolare, l'accuracy complessiva parte da 0.26 alla seconda epoca e si mantiene costante a 0.28 nelle epoche successive. Questo andamento suggerisce che BERT riesce a preservare efficacemente le conoscenze linguistiche acquisite durante il pre-addestramento, confermando la sua robustezza per task su dati italiani standard. Inoltre, la stabilità dei risultati indica una minore predisposizione all'overfitting rispetto a quanto osservato per Minerva.

L'elevata capacità di conservare le conoscenze linguistiche acquisite durante il pre-addestramento, è, altresì, evidenziata dai test eseguiti su una frase di esempio standard. Utilizzando il task di Masked Language Modeling (MLM) con l'input "La capitale dell'Italia è [MASK].", il modello ha sistematicamente identificato correttamente Roma come risposta più probabile in tutte le epoche di addestramento, con leggere variazioni nello score.

Ad esempio, dopo 2 epoche di fine-tuning, il modello ha restituito le predizioni seguenti:

1. *roma* (score: 0.6016)
2. *milano* (score: 0.0923)
3. *vasto* (score: 0.0226)
4. *napoli* (score: 0.0165)
5. *firenze* (score: 0.0114)

Dopo 6 epoche, l'output rimane coerente, indicando Roma come la parola più probabile, con uno score leggermente inferiore (0.5195). Questo risultato conferma che il modello con-

serva le conoscenze pregresse acquisite durante il pre-addestramento, evitando significative perdite di generalizzazione anche dopo il fine-tuning.

4.3 Risultati TWS con BERT

La valutazione del Target Word Selection (TWS) con BERT ha fornito risultati estremamente promettenti, dimostrando la capacità del modello di eccellere in questo task una volta implementata una strategia adeguata di pre-processing e di data collation. I risultati ottenuti, riportati nella Tabella 4.6, evidenziano performance superiori sia sul test set in dialetto che sui dati italiani e dialettali del training.

metriche	**test dialetto**	**train dialetto**	**dataset italiano**
Accuracy complessiva	0.76	0.90	0.82
Top-3 Accuracy	0.98	0.99	0.96
Similarità media	0.83	0.93	0.87

Tabella 4.6: Risultati della valutazione del TWS

Inizialmente, è stato testato l'utilizzo del *DataCollatorForLanguageModeling* di Hugging Face, lo stesso utilizzato per il task TWP. Tuttavia, i risultati iniziali erano insoddisfacenti in quanto il modello otteneva un'accuracy del 25%, equivalente a una selezione casuale tra le quattro opzioni. Questo risultato evidenziava la necessità di personalizzare il data collator per adattarlo al task TWS. Questo approccio ha permesso di strutturare correttamente il dataset in dialetto per il fine-tuning e gli input del modello per il task di scelta multipla, rispettando la natura del TWS e migliorando sensibilmente le performance.

Sul test set in dialetto, il modello ha raggiunto un'accuracy complessiva del 76%, mentre sui dati di training e sul dataset italiano ha ottenuto, rispettivamente, il 90% e l'82%.

La Top-3 accuracy è estremamente elevata in tutte le configurazioni, sfiorando il 98% sul test set e raggiungendo il 99% sul training set. Questi valori confermano che, anche quando non individua l'opzione corretta come prima scelta, il modello include la risposta giusta tra le prime tre opzioni.

I risultati complessivamente evidenziano la capacità di BERT di generalizzare efficacemente il task TWS su dati dialettali, nonostante l'addestramento limitato.

L'alta accuracy sul training set in dialetto indica che il modello è stato in grado di apprendere efficacemente dai dati forniti, riuscendo ad identificare la parola corretta con grande precisione.

L'accuracy sul test set in dialetto dimostra una buona capacità di generalizzazione del modello, sebbene evidenzi un margine di miglioramento rispetto ai dati di training.

Sul dataset italiano, il modello ha ottenuto un'accuracy complessiva dell'82%, confermando la robustezza di BERT nel mantenere le conoscenze acquisite durante il pre-addestramento anche per task più specifici.

In conclusione, il task TWS rappresenta un esempio riuscito di approccio alla comprensione e modellazione del dialetto. Questo risultato dimostra come la combinazione di strategie adattive e un fine-tuning specifico possa generare risultati concreti e significativi, anche in contesti linguistici complessi e caratterizzati da risorse limitate.

CAPITOLO 5

Discussione

In questo capitolo si discuteranno i risultati ottenuti durante la fase di sviluppo e sperimentazione dei Large Language Model applicati al dialetto fermano. La discussione inizierà con un'analisi approfondita dei risultati, in cui verranno esaminati i vari aspetti delle prestazioni dei modelli Minerva e BERT, confrontando i risultati ottenuti con diverse epoche di addestramento e sui vari dataset, per evidenziare i punti di forza e le limitazioni di ciascun approccio. Dopo di ciò, verranno esplorate le problematiche riscontrate, che includono sfide sia tecniche che concettuali. Infine, il capitolo si concluderà con una panoramica delle prospettive di ottimizzazione a breve termine che riguardano interventi specifici con l'obiettivo di migliorare le prestazioni complessive e affrontare le sfide emerse durante l'intero processo di sviluppo.

5.1 Analisi dei risultati

L'analisi critica dei risultati ottenuti dai modelli Minerva e BERT evidenzia tanto i progressi quanto le limitazioni delle tecniche e dei metodi applicati.

La valutazione qualitativa del task di generazione del testo ha evidenziato limiti significativi in Minerva, specialmente in contesti linguistici complessi come il dialetto fermano. Sebbene sia stato osservato un lieve miglioramento con l'aumentare delle epoche di addestramento, il modello ha mostrato difficoltà persistenti nel produrre testo coerente e fluido. I punteggi per adeguatezza e fluidità, riportati graficamente nella Figura 5.1, riflettono una chiara incapacità di rispondere in modo pertinente e grammaticalmente corretto persino ad input semplici come "La capitale dell'Italia è".

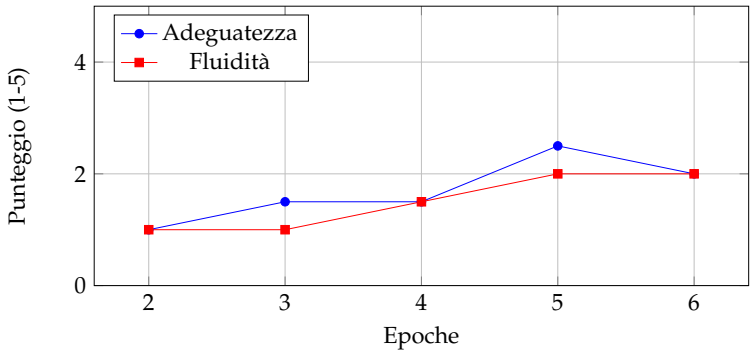


Figura 5.1: Valutazione qualitativa della generazione di testo

Un aspetto cruciale emerso dai risultati riguarda la difficoltà del modello a preservare le conoscenze pregresse acquisite durante il pre-addestramento. Nonostante l'obiettivo del fine-tuning fosse quello di adattare Minerva al dialetto fermano, i risultati indicano che il modello ha sacrificato parte delle competenze linguistiche di base e delle conoscenze generali, producendo frasi che mancano di coerenza e struttura. Questa perdita di conoscenze pregresse è particolarmente evidente nelle risposte incoerenti o artificiose generate dal modello, come dimostrano gli output delle ultime epoche, dove l'inserimento di termini dialettali avviene in modo sconnesso e non contestualizzato.

Un altro punto di riflessione riguarda l'impatto del fine-tuning sul comportamento generale del modello. L'adattamento al dialetto, pur avendo introdotto elementi linguistici plausibili dal punto di vista locale, sembra aver compromesso ulteriormente la capacità del modello di generare testo grammaticalmente corretto. Sebbene nelle epoche intermedie si siano osservati tentativi più coerenti di completare la frase iniziale con riferimenti geografici, questi risultati sono stati sporadici e non rappresentano un progresso uniforme. Questo comportamento suggerisce che il modello sta apprendendo caratteristiche superficiali dai dati dialettali piuttosto che una comprensione profonda delle strutture linguistiche del dialetto.

In sintesi, i risultati qualitativi sottolineano le sfide associate all'adattamento di un modello linguistico generale a un contesto altamente specifico come il dialetto fermano. Sebbene il fine-tuning abbia introdotto miglioramenti parziali, essi non sono stati sufficienti a superare le limitazioni strutturali del modello. Tali risultati evidenziano la necessità di strategie di ottimizzazione delle metodologie adottate.

La valutazione pre fine-tuning del task di Target Word Prediction è riportata graficamente nella Figura 5.2.

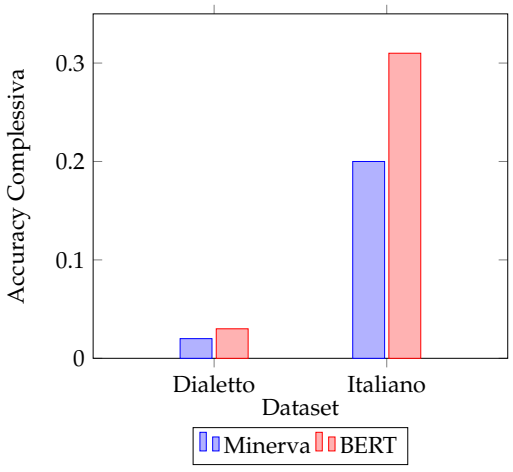


Figura 5.2: Confronto dell'accuracy complessiva del TWP pre fine-tuning

Tali risultati hanno confermato che i modelli Minerva e BERT, nella loro versione pre-addestrata, non sono adatti per il contesto dialettale. Questo era un risultato prevedibile, considerando che nessuno dei due modelli è stato addestrato su dati specifici del dialetto fermano. Questi modelli, progettati per la lingua italiana standard, falliscono completamente nel gestire un dominio linguistico così distante dalle loro conoscenze preesistenti. Pertanto, era ragionevole aspettarsi che il fine-tuning potesse colmare almeno in parte questa lacuna.

Diverso è il caso del dataset italiano. Considerando che entrambi i modelli sono stati pre-addestrati su grandi quantità di dati in italiano, le aspettative sulle prestazioni erano significativamente più alte. Tuttavia, i risultati ottenuti mostrano un quadro meno positivo del previsto. Questa discrepanza rispetto alle aspettative può essere attribuita alla tipologia

del task. Il task TWP, infatti, è essenzialmente un task di completamento di una frase mentre BERT dimostra una maggiore capacità nel prevedere token all'interno di una sequenza in quanto può sfruttare la comprensione del contesto in modo bidirezionale. Inoltre, il dataset italiano utilizzato per la valutazione potrebbe rappresentare una sfida anche per modelli pre-addestrati su dati italiani generici. La specificità del linguaggio e delle strutture semantiche presenti in questi riassunti potrebbe non essere perfettamente in linea con i dati su cui Minerva e BERT sono stati pre-addestrati.

In conclusione, il fine-tuning ha comunque portato miglioramenti significativi sul dialetto, dato il punto di partenza così basso. Sul dataset italiano, tuttavia, il margine di miglioramento è stato più limitato, sia per la natura del task che per il livello di prestazioni già raggiunto dai modelli pre-addestrati.

I grafici presentati in Figura 5.3 mostrano un confronto diretto tra Minerva e BERT fine-tuned relativamente all'accuracy complessiva sui tre dataset testati. Questa comparazione mette in luce dinamiche complesse tra i due modelli che si differenziano significativamente per prestazioni e comportamento durante il fine-tuning.

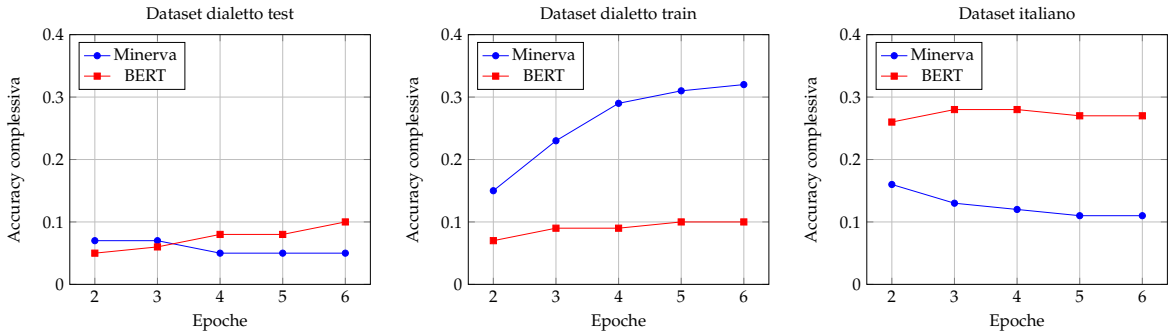


Figura 5.3: Confronto dell'accuracy complessiva del TWP con Minerva e BERT

I risultati ottenuti indicano che Minerva ha difficoltà a mantenere un equilibrio tra apprendimento e generalizzazione. Sul dataset dialettale di test, l'accuracy complessiva si stabilizza già dalle prime epoche, suggerendo che il modello raggiunge un limite nelle sue capacità di apprendimento per nuovi contesti. Tuttavia, sul dataset dialettale di train, si osserva un aumento costante dell'accuracy, segno evidente di overfitting. Questo fenomeno è ulteriormente confermato dai risultati sul dataset italiano, dove l'accuracy complessiva diminuisce con l'aumentare delle epoche, suggerendo una perdita delle conoscenze linguistiche pregresse acquisite nel pre-addestramento.

Viceversa, BERT dimostra una maggiore robustezza. Sul dataset italiano, infatti, l'accuracy rimane stabile tra le epoche, indicando che il modello riesce a preservare le sue competenze di base. Tuttavia, i miglioramenti sul dataset dialettale di train sono più contenuti rispetto a Minerva. Questo suggerisce una minore capacità di apprendimento e adattamento a nuovi contesti linguistici.

La differenza tra le prestazioni dei due modelli sottolinea come le rispettive architetture influiscano sulla loro capacità di apprendere e generalizzare. Questi risultati mettono in luce una dinamica complessa tra flessibilità e conservazione delle conoscenze pregresse. Minerva dimostra un'ottima capacità di adattamento ai dati dialettali ma a scapito delle conoscenze precedenti. BERT, al contrario, mostra una maggiore resistenza all'overfitting, mantenendo inalterate le competenze sulla lingua italiana, ma con una capacità limitata di apprendere nuove informazioni sui dati dialettali.

Il task di Target Word Selection (TWS) si è rivelato un caso di successo significativo nell'intero lavoro di fine-tuning e valutazione, evidenziando capacità sorprendenti da parte di BERT nel contesto dialettale e italiano.

A differenza del TWP, il task TWS consente al modello di operare in un ambiente più guidato. La presenza di un insieme predefinito di opzioni riduce, infatti, la complessità del problema, restringendo lo spazio delle risposte possibili. Questo potrebbe spiegare perché le performance siano significativamente più elevate rispetto al TWP. Accuracy complessiva, top-3 accuracy e similarità media raggiungono valori rispettabili sia nel dialetto che nell'italiano. Tuttavia, l'apparente "facilità" del task potrebbe non riflettere interamente la capacità di comprensione linguistica del modello, ma piuttosto la sua abilità a fare inferenze basate sulle opzioni disponibili.

Nonostante le elevate performance del TWS, il task presenta limiti concettuali. Il successo dipende in gran parte dalla qualità delle opzioni generate: se le opzioni includessero termini semanticamente meno correlati al contesto, l'accuratezza potrebbe diminuire drasticamente. Questo evidenzia come il task non sia una misura assoluta della comprensione linguistica, ma piuttosto un indicatore della capacità del modello di distinguere la parola più appropriata tra un insieme ristretto di possibilità. Tale task ha dimostrato la capacità di BERT a riconoscere e apprendere relazioni tra parole anche in un dialetto poco rappresentato nei dati di pre-addestramento.

5.2 Problematiche riscontrate

Lo sviluppo e la valutazione del modello hanno portato alla luce diverse problematiche tecniche e concettuali che hanno influenzato sia l'implementazione della soluzione che le performance finali dei modelli. Queste difficoltà sottolineano la complessità nell'adattare modelli linguistici pre-addestrati a un contesto altamente specifico come quello di un dialetto marchigiano, evidenziando sfide su più fronti.

Di seguito, vengono analizzati i principali limiti emersi durante il progetto:

- *Limiti dei dataset disponibili*: uno dei problemi più significativi è stata la scarsità di dati in dialetto fermano. La costruzione del dataset ha richiesto un lavoro intenso di raccolta, pulizia e segmentazione di testi derivati principalmente da fonti non digitalizzate, come commedie e poesie dialettali. La mancanza di una quantità sufficiente di dati ha limitato il potenziale del fine-tuning, influenzando negativamente la capacità dei modelli di generalizzare. Inoltre, i dati disponibili spesso presentavano incongruenze linguistiche dovute alla natura orale del dialetto e alla sua mancanza di standardizzazione. Ciò ha reso difficile stabilire regole coerenti per la tokenizzazione e il pre-processing, introducendo variabilità indesiderata.

- *Limitazioni metodologiche nella valutazione*: le metriche utilizzate, sebbene utili per valutazioni quantitative, non sono state sufficienti a catturare tutte le sfumature della generazione di testo in dialetto. Ad esempio, la similarità del coseno calcolata con Sentence-BERT ha mostrato limiti nel confrontare parole dialettali con variazioni morfologiche non standard. La valutazione qualitativa, inoltre, sebbene efficace nel fornire insight sull'adeguatezza e la fluidità del testo generato, è risultata soggettiva e dipendente dall'interpretazione del valutatore. Questo ha evidenziato la necessità di sviluppare metodi di valutazione più specifici per il contesto dialettale.

- *Complessità computazionale*: il fine-tuning di modelli di grandi dimensioni come Minerva e BERT ha richiesto risorse computazionali significative, limitando la possibilità di sperimentare con un maggior numero di epoche o di configurazioni di iperparametri. Inoltre, a causa di tale problematica non è stato possibile confrontare ulteriori Large Language Model più grandi e potenti che potevano comportare performance migliori.

- *Adattamento del dataset al TWS*: il task di Target Word Selection ha richiesto, oltre ad un Data Collator personalizzato, la ristrutturazione dei dataset, quindi una pipeline di pre-processing specifica, per rendere coerenti i dati in input al modello e garantire risultati validi.
- *Mancanza di benchmark dialettali*: l'assenza di benchmark preesistenti per il dialetto fermano ha reso difficile collocare i risultati in un contesto più ampio. La mancanza di dati comparativi ha limitato la capacità di comprendere globalmente i punti di forza e le debolezze dei modelli.

5.3 Prospettive di ottimizzazione

Per affrontare le problematiche emerse e migliorare i risultati ottenuti, si propongono alcune strategie di ottimizzazione a breve termine, suddivise in ambiti specifici.

Un primo passo fondamentale è l'ampliamento del dataset dialettale. La raccolta di nuove fonti, come registrazioni audio trascritte di dialoghi in dialetto o documenti storici, potrebbe aumentare la rappresentatività dei dati. Inoltre, l'uso di tecniche di data augmentation, come la traduzione inversa (ovvero tradurre una frase in un'altra lingua e poi ritradurla nel dialetto fermano), il Synonym replacement (cioè sostituire parole chiave con sinonimi dialettali) o la perturbazione fonetica (vale a dire simulare variazioni ortografiche per rappresentare differenze regionali o di pronuncia), potrebbe migliorare la varietà del dataset senza introdurre errori significativi.

L'impiego di Large Language Models di ultima generazione, come LLaMA, potrebbe garantire capacità di generalizzazione e adattamento superiori rispetto a modelli come Minerva e BERT. Questi modelli, grazie alla loro maggiore capacità e ricercata architettura, potrebbero gestire meglio la variabilità linguistica del dialetto fermano, anche in presenza di dati limitati. Tuttavia, il loro utilizzo richiede risorse computazionali considerevoli, il che sottolinea la necessità di infrastrutture adeguate per sfruttare appieno le loro potenzialità.

Un altro approccio innovativo è l'introduzione di tecniche di Continual Learning, un approccio progettato per consentire ai modelli di apprendere progressivamente nuovi compiti senza dimenticare quanto appreso in precedenza. Questo paradigma sarebbe particolarmente utile per preservare le conoscenze pregresse, come quelle sul dataset italiano standard, mentre si adatta il modello a nuovi domini, come il dialetto.

Tra i metodi di Continual Learning, l'Elastic Weight Consolidation (EWC) è particolarmente rilevante. L'EWC aggiunge un termine di regolarizzazione alla funzione di perdita del modello, vincolando i pesi più importanti affinché non subiscano variazioni significative durante l'apprendimento di nuovi compiti. Questa tecnica aiuta a preservare le competenze linguistiche di base apprese nel pre-addestramento, mitigando il fenomeno del "catastrophic forgetting". Applicare l'EWC potrebbe mantenere un bilanciamento tra adattamento al dialetto e conservazione delle conoscenze linguistiche di base.

Un approccio promettente per migliorare la generazione di testo in dialetto potrebbe essere l'adozione del fine-tuning condizionale, una tecnica che specializza il modello nel rispondere in modo mirato a specifici tipi di input. Rispetto al fine-tuning generico di language modeling, che addestra il modello sulla probabilità di sequenze di testo senza distinzione del contesto, il fine-tuning condizionale introduce un meccanismo in cui il modello è guidato da un input esplicito, spesso chiamato "condizione" o "prompt". Per il dialetto fermano, il fine-tuning condizionale potrebbe consistere nell'addestrare il modello a generare testo basato su specifiche condizioni legate a un contesto culturale, storico o linguistico del dialetto come la traduzione di una frase. Ad esempio, nel caso del task di Target Word Prediction, il modello potrebbe essere addestrato a completare frasi in dialetto fornendo

un contesto esplicito come: "Completa la frase dialettale seguente". Questo approccio è ampiamente utilizzato per LLM come GPT, ChatGPT o LLaMA, per attività che includono rispondere a domande dirette, generare suggerimenti, storie, poesie o descrizioni, o eseguire compiti come riassumere e tradurre. Il modello impara, quindi, non solo a completare o generare frasi, ma a rispondere a richieste che seguono uno schema logico definito.

La configurazione degli iperparametri svolge un ruolo cruciale nell'efficacia dell'addestramento. Parametri come il learning rate, la probabilità di masking e la dimensione del batch influenzano direttamente la convergenza del modello e la sua capacità di apprendimento. L'adozione di tecniche di ricerca automatica, come la grid search o la random search, permetterebbe di identificare configurazioni ottimali in modo sistematico. Questo potrebbe migliorare sia l'efficienza dell'addestramento sia le performance finali, minimizzando il rischio di overfitting.

Un ulteriore passo avanti potrebbe essere compiuto nello sviluppo di metriche di valutazione più specifiche e aderenti alle caratteristiche del dialetto. L'uso delle tradizionali metriche di valutazione, seppur utili, non sempre rispecchia adeguatamente le peculiarità linguistiche del dialetto, come le variazioni morfologiche e semantiche, che possono influenzare la comprensione e la generazione del testo. Ad esempio, l'uso delle Diversity Metrics potrebbe essere particolarmente utile per valutare la varietà e la creatività del testo generato dal modello, misurando la presenza di n-grammi unici o la diversità lessicale nel dialetto. Questo approccio permetterebbe di evitare ripetizioni e risposte troppo standardizzate, caratteristiche che potrebbero non rispecchiare la vivacità del dialetto. Inoltre, un altro strumento utile potrebbe essere l'Adversarial Evaluation, tramite modelli che valutano la naturalità del testo generato rispetto a quello umano. In questo caso, un modello avversario potrebbe distinguere il testo generato da un modello da quello scritto da un parlante umano, contribuendo a misurare quanto il modello riesca a produrre risposte che suonano autentiche e naturali. Pertanto, la progettazione di metriche che tengano conto di queste differenze linguistiche potrebbe offrire una valutazione più precisa delle capacità del modello, migliorando la comprensione del suo livello di competenza linguistica nel dialetto specifico.

Infine, l'adozione di infrastrutture distribuite, come l'utilizzo di multi-GPU o TPU su piattaforme cloud (ad esempio AWS, Google Cloud o Azure), rappresenta una soluzione efficace per affrontare la complessità computazionale del fine-tuning di modelli di grandi dimensioni. Queste infrastrutture consentono di parallelizzare i carichi di lavoro, riducendo significativamente i tempi di addestramento e permettendo di esplorare configurazioni di iperparametri o architetture più avanzate in maniera più efficiente.

Queste prospettive di ottimizzazione rappresentano passi pratici e realistici per migliorare i risultati e ampliare le potenzialità del sistema sviluppato. Sebbene alcune di queste strategie richiedano risorse e competenze aggiuntive, i benefici attesi potrebbero giustificare l'investimento e portare ad un sistema di generazione di testo dialettale più robusto ed efficace.

Conclusioni

La presente tesi ha esplorato l'adattamento di Large Language Model pre-addestrati, come Minerva e BERT, a un contesto linguistico unico e scarsamente rappresentato, ovvero il dialetto fermano. Il lavoro è stato sviluppato lungo un percorso articolato, suddiviso in più fasi.

Inizialmente, è stato analizzato il panorama dei Large Language Model, descrivendone i principi di funzionamento, le architetture principali e le applicazioni, con un focus sulle potenzialità e i limiti che questi modelli presentano. Successivamente sono stati definiti i requisiti del sistema, con particolare attenzione alle risorse linguistiche necessarie e alle sfide tecniche e concettuali legate all'addestramento e alla valutazione dei modelli.

La fase di implementazione ha previsto la raccolta e la preparazione dei dati in dialetto fermano, un'operazione complessa che ha richiesto l'acquisizione di fonti non digitalizzate, la loro pulizia e segmentazione, e la creazione di un dataset utilizzabile per il fine-tuning. Sono stati definiti tre task distinti per la valutazione delle capacità dei modelli, ovvero generazione di testo, Target Word Prediction (TWP) e Target Word Selection (TWS). Ognuno di essi ha richiesto un approccio specifico, dall'adattamento dei dati alla personalizzazione di pipeline di pre-processing e metriche di valutazione.

I risultati ottenuti sono stati analizzati e discussi al fine di mettere in luce le potenzialità e i limiti dei modelli linguistici esaminati nel comprendere e generare testo in dialetto. Minerva ha dimostrato una maggiore capacità di adattamento sul dialetto rispetto a BERT, seppur a scapito della preservazione delle conoscenze pregresse. D'altra parte, BERT ha mostrato una maggiore robustezza nel mantenere le competenze linguistiche di base, ma una minore efficacia nell'apprendere nuove informazioni dal dialetto. Il task di TWS si è rivelato particolarmente promettente, con risultati solidi anche in un contesto dialettale. Tuttavia, la generazione di testo e il TWP hanno evidenziato significative aree di miglioramento, soprattutto in termini di coerenza e fluidità.

Infine, sono state analizzate in dettaglio le sfide, le limitazioni e le difficoltà incontrate durante la progettazione e l'implementazione delle soluzioni proposte. Questi aspetti hanno messo in luce le complessità intrinseche nell'adattare modelli linguistici pre-addestrati a un contesto linguistico così specifico e sottorappresentato. Successivamente, sono state esplorate prospettive di ottimizzazione mirate, volte a superare tali ostacoli e a migliorare le prestazioni dei modelli, proponendo strategie concrete per affinare i risultati ottenuti e ampliare le potenzialità del lavoro svolto.

L'adattamento di modelli linguistici pre-addestrati al dialetto fermano rappresenta un esempio pionieristico nel campo dell'elaborazione del linguaggio naturale, evidenziando le sfide uniche e i potenziali benefici di estendere tali metodologie a contesti linguistici

sottorappresentati. Tuttavia, il lavoro non si esaurisce nei risultati ottenuti. Al contrario, esso pone le basi per sviluppi futuri che potrebbero ampliare significativamente l'impatto di questi modelli, non solo nel contesto dialettale, ma anche in altri settori culturali, sociali e tecnologici.

Un primo sviluppo futuro consisterebbe nell'adattare le metodologie e gli approcci sviluppati ad altri dialetti italiani o lingue regionali sottorappresentate. L'Italia, con la sua ricchezza linguistica, offre un vasto campo d'applicazione per preservare e valorizzare il patrimonio culturale delle varietà dialettali. Analogamente, l'approccio potrebbe essere esteso a lingue minoritarie in altre parti del mondo, contribuendo a colmare il divario nella rappresentazione linguistica nei modelli di Intelligenza Artificiale.

Un altro sviluppo significativo potrebbe essere la creazione di chatbot o assistenti vocali basati sui modelli addestrati in dialetto fermano. Questi strumenti potrebbero essere utilizzati in contesti turistici ma anche per favorire il dialogo intergenerazionale, coinvolgendo le persone anziane che parlano quotidianamente il dialetto negli strumenti che offre l'Intelligenza Artificiale.

Una prospettiva interessante sarebbe l'integrazione dei modelli linguistici con sistemi multimodali, che combinano testo, immagini e audio. Ad esempio, un sistema capace di generare descrizioni in dialetto fermano a partire da immagini o video potrebbe trovare applicazione in musei, mostre o tour virtuali.

In conclusione, il lavoro svolto rappresenta un punto di partenza importante per l'adattamento di modelli linguistici a contesti caratterizzati da scarsità di risorse. Sebbene le sfide siano molteplici, le potenziali applicazioni di questi strumenti sono vastissime, offrendo nuove opportunità per preservare e valorizzare il patrimonio linguistico e culturale in modi innovativi e accessibili.

Bibliografia

AL SHURAIQI, S., AAL ABDULSALAM, A., MASTERS, K., ZIDOUM, H. e ALZAABI, A. (2024), «Automatic Generation of Medical Case-Based Multiple-Choice Questions (MCQs): A Review of Methodologies, Applications, Evaluation, and Future Directions», *Big Data and Cognitive Computing*, vol. 8.

ALAM, F., CHOWDHURY, S. A., BOUGHORBEL, S. e HASANAIN, M. (2024), «LLMs for Low Resource Languages in Multilingual, Multimodal and Dialectal Settings», in «Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts», p. 27–33, Association for Computational Linguistics.

DEVLIN, J., CHANG, M.-W., LEE, K. e TOUTANOVA, K. (2018), «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding», *arXiv preprint arXiv:1810.04805*.

GOLDBERG, Y. (2017), *Neural Network Methods for Natural Language Processing*, Morgan & Claypool.

JOSHI, A., DABRE, R., KANOJIA, D., LI, Z., ZHAN, H., HAFFARI, G. e DIPPOLD, D. (2024), «Natural Language Processing for Dialects of a Language: A Survey», *arXiv preprint arXiv:2401.05632*.

JURAFSKY, D. e MARTIN, J. H. (2021), *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Pearson, 3rd ed.

KAMATH, U., KEENAN, K., SOMERS, G. e SORENSON, S. (2024), *Large Language Models: A Deep Dive: Bridging Theory and Practice*, Springer.

KANTHARUBAN, A., VULIĆ, I. e KORHONEN, A. (2023), «Quantifying the Dialect Gap and its Correlates Across Languages», *arXiv preprint arXiv:2310.15135*.

MINAEE, S., MIKOLOV, T., NIKZAD, N., CHENAGHLU, M., SOCHER, R., AMATRIAIN, X. e GAO, J. (2024), «Large Language Models: A Survey», *arXiv pre print arXiv:2402.06196*.

ONDREJOVÁ, V. e ŠUPPA, M. (2024), «Can LLMs Handle Low-Resource Dialects? A Case Study on Translation and Common Sense Reasoning in Šariš», in «Proceedings of the Eleventh Workshop on NLP for Similar Languages, Varieties, and Dialects (VarDial 2024)», p. 130–139.

PATWARDHAN, N., MARRONE, S. e SANSONE, C. (2023), «Transformers in the Real World: A Survey on NLP Applications», *Information*, vol. 14.

RADFORD, A., NARASIMHAN, K., SALIMANS, T. e SUTSKEVER, I. (2018), «Improving Language Understanding by Generative Pre-Training», *OpenAI*.

REIMERS, N. e GUREVYCH, I. (2019), «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks», *arXiv preprint arXiv:1908.10084*.

SRIRAG, D., SAHOO, N. R. e JOSHI, A. (2024), «Evaluating Dialect Robustness of Language Models via Conversation Understanding», *arXiv pre print arXiv:2405.05688*.

TUNSTALL, L. e LEANDRO VON WERRA AND, T. W. (2022), *Natural Language Processing with Transformer*, O'Reilly Media.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L. e POLOSUKHIN, I. (2017), «Attention Is All You Need», *Advances in Neural Information Processing Systems*, vol. 30.

WANG, Z., CHU, Z., DOAN, T. V., NI, S., YANG, M. e ZHANG, W. (2024), «History, development, and principles of large language models: an introductory survey», *arXiv pre print arXiv:2402.06853*.

XU, Y., LI, M., DING, N., DOU, Z. e CHENG, X. (2021), «Adapting Language Models to Dialects via Few-Shot Learning», *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

Sitografia

- ACL Anthology – www.aclanthology.org
- Arxiv – www.arxiv.org
- DeepLearning.AI – www.deeplearning.ai
- GitHub – www.github.com
- Google Scholar – www.scholar.google.com
- Hugging Face, The AI community building the future – www.huggingface.com
- Medium – www.medium.com
- OpenAI – www.openai.com
- ResearchGate – www.researchgate.net
- Spacy – spacy.io
- Stack Overflow – www.stackoverflow.com
- TensorFlow –www.tensorflow.org
- Treccani – www.treccani.it
- Wikipedia – www.wikipedia.org

Ringraziamenti

Devo ammettere che questa è la parte più difficile da scrivere di tutta la tesi. Parlare di ciò che si studia e che risiede nella testa è decisamente più semplice di esprimere ciò che dimora nel cuore. Per questo sarò molto breve deludendo, probabilmente, le vostre aspettative nel farvi commuovere ma risparmiandovi il solito ringraziamento cringe per avermi supportato e sopportato.

In primis, un ringraziamento doveroso va al mio relatore, il Prof. Ursino, che, insieme ai suoi collaboratori Davide, Luca e Michele, mi ha guidato con competenza e disponibilità nell'ultima curva di questo lungo percorso.

La dedica più sentita va, ovviamente, alla mia famiglia. Due parole scritte su un foglio non possono esprimere e ripagare tutto ciò che avete fatto e fate per me. Per questo mi limito a dirvi grazie, nel modo più autentico e devoto. Spero, un giorno, di riuscire a ricambiare, non solo con i miei successi, tutti i vostri sacrifici. Nel frattempo sappiate che ogni traguardo raggiunto è anche il vostro.

Vorrei ringraziare Andrea semplicemente citando ciò che ha detto Roberto Benigni alla sua amata: "Conosco una solo maniera di misurare il tempo: con te e senza di te.". La tua presenza nella mia vita è la mia più grande felicità.

Un grazie va a tutti miei amici, quelli con cui sono cresciuta, quelli conosciuti durante l'adolescenza o più tardi, quelli con cui ho condiviso questo percorso universitario e quelli con cui ho convissuto. Vi ringrazio perché riconosco che non è facile instaurare un legame con una persona così chiusa e anaffettiva come me ma voi tutti siete riusciti ad occupare uno spazio nel mio cuore creando un puzzle perfetto.

Ora è il momento dell'autocelebrazione che mi risulta sempre molto semplice. Ce l'hai fatta Laura, ce l'hai fatta a combattere la tua più grande nemica: te stessa. Non sentirti mai più sbagliata, mai più inferiore ad altri. Continua a dare sempre il massimo, sii felice e facci caso. Ricorda che la vita è fatta di luci e ombre: se poggi l'attenzione sulla luce riesci a gestire con serenità anche le ombre; se poggi l'attenzione solamente sulle ombre della luce neanche ti accorgi ed è un grande spreco.