Relazione progetto BERT

Corso di Data Science

A.A. 2023/2024

Docenti:
Prof. Domenico Ursino
Dott. Michele Marchetti

Studenti:
Civitarese Andrea
Longarini Lorenzo
Pasquini Gioele
Ramovini Loris

Sommario

Indice delle figure

1 Introduzione

Questo progetto verte sul fine tuning del modello pre-addestrato BERT al fine di classificare recensioni di film rilasciate sul sito IMDB tramite NLP.
Abbiamo utilizzato Google Colab per effettuare il fine tuning poiché, essendo un procedimento abbastanza pesante e richiedendo risorse GPU per essere velocizzato, i nostri computer non erano in grado di eseguirlo in breve tempo. Questa piattaforma mette a disposizione un certo quantitativo di risorse di calcolo e sfrutta la tecnologia Jupyter Notebook per eseguire codice Python.
Il codice utilizzato per il progetto è visualizzabile su GitHub al seguente link.

1.1 Cos'è l'NLP

L'NLP, acronimo di Natural Language Processing, è un'area dell'informatica che si occupa dei metodi per analizzare, modellare e comprendere il linguaggio umano. L'obiettivo principale del NLP è quello di consentire alle macchine di interagire con gli esseri umani in modo più naturale e intelligente possibile, elaborando e interpretando il testo e il linguaggio parlato.
Le applicazioni del NLP sono ampie e diversificate. Tra queste ci sono:

- **Elaborazione del linguaggio naturale**: analisi e comprensione del testo per estrarre informazioni significative, come sentimenti, entità e relazioni;
- **Traduzione automatica**: conversione automatica di testo da una lingua all'altra;
- **Generazione di testo**: creazione automatica di testo coerente e comprensibile;
- **Risposte automatiche**: sviluppo di sistemi che possono rispondere a domande poste in linguaggio naturale;
- **Ricerca e recupero di informazioni**: estrazione di informazioni rilevanti da grandi quantità di testo;
- **Analisi dei sentimenti**: valutazione del tono emotivo e delle opinioni espressi nel testo.
- **Sintesi vocale**: conversione del testo in parlato.

Il NLP utilizza una varietà di tecniche e algoritmi, tra cui modelli statistici, deep neural networks, algoritmi di apprendimento automatico e regole linguistiche. Negli ultimi anni, il NLP ha fatto enormi progressi grazie allo sviluppo di modelli di linguaggio basati su trasformatori, come BERT, GPT e XLNet, che hanno portato a risultati più precisi e versatili in molte applicazioni di elaborazione del linguaggio naturale.

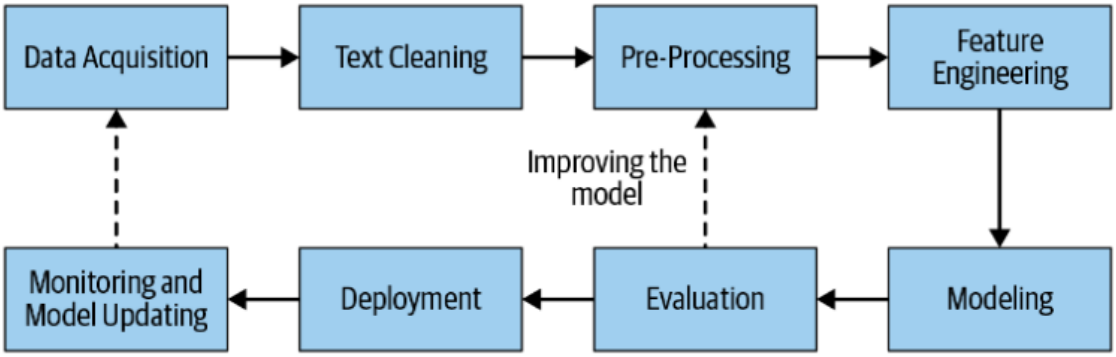

Figura 1: Pipeline NLP

La pipeline di elaborazione del linguaggio naturale (NLP) è un flusso di lavoro che comprende una serie di passaggi sequenziali per creare un modello che riesca ad analizzare e comprendere il testo. Questi passaggi possono variare a seconda dell'applicazione specifica, ma solitamente includono le seguenti fasi:

1. **Acquisizione dei dati**: Il testo, necessario per l'addestramento del modello, viene acquisito da diverse fonti come, ad esempio, siti internet, dataset preesistenti e libri digitali;
2. **Pulizia dei dati:** Rimozione di caratteri speciali, punteggiatura, e altri elementi non rilevanti che potrebbero interferire con l'analisi successiva;
3. **Pre-Processing**: Serie di procedure come tokenizzazione, rimozione di punteggiatura o altro, è molto variabile a seconda dell'applicazione;
4. **Feature engineering**: Cattura delle caratteristiche semantiche di una parola in un vettore numerico che sia comprensibile dai modelli automatici;
5. **Modeling**: Scelta e creazione del modello in base ai dati a disposizione e all'applicazione da realizzare, si va dalle semplici regole euristiche a complesse reti neurali;
6. **Evaluation**: Verifica delle prestazioni del modello creato;
7. **Deployment**: Rilascio del modello e integrazione nel sistema di lavoro finale;
8. **Monitoring and Updating**: Monitoraggio delle prestazioni del modello e ulteriore fine tuning se ci sono dati disponibili e se necessario.

1.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) è un modello di linguaggio basato su trasformer sviluppato da Google. Lanciato nel 2018, BERT ha rivoluzionato il campo dell'elaborazione del linguaggio naturale grazie alla sua capacità di comprendere il contesto e la relazione delle parole all'interno di una frase in modo bidirezionale.
A differenza dei modelli precedenti che trattavano il testo sequenzialmente e in una sola direzione (da sinistra a destra o viceversa), BERT utilizza un'architettura bidirezionale che considera il contesto sia a sinistra che a destra di ciascuna parola durante l'elaborazione. Questo permette a BERT di catturare relazioni più complesse e semantiche tra le parole, migliorando significativamente la sua capacità di comprensione del linguaggio umano.
BERT viene pre-addestrato su grandi quantità di testo non annotato utilizzando compiti di apprendimento automatico auto-supervisionato, come la previsione di parole mancanti in una frase o la classificazione di frasi successive. Questo processo di pre-training consente a BERT di apprendere rappresentazioni linguistiche di alto livello che possono essere trasferite e adattate a una vasta gamma di compiti NLP.
Grazie alla sua versatilità e alle prestazioni superiori rispetto ai modelli precedenti, BERT è diventato uno degli strumenti più utilizzati e influenti nel campo del NLP. È stato adottato da molte aziende e ricercatori per migliorare le prestazioni dei propri sistemi NLP in una varietà di applicazioni e settori.
In particolare, riguardo l'addestramento di un modello basato su BERT, questo viene effettuato in due step:

1. **Pre-addestramento**: In questa fase, BERT viene addestrato su grandi quantità di testo non annotato. Durante il Pre-addestramento, il modello affronta due compiti principali:
    - **Masked Language Model (MLM):** BERT maschera casualmente alcune parole nel testo di input e cerca di prevederle utilizzando il contesto circostante. Questo compito forza BERT a comprendere il significato delle parole in relazione alle parole circostanti e migliora la sua capacità di catturare il contesto bidirezionale.

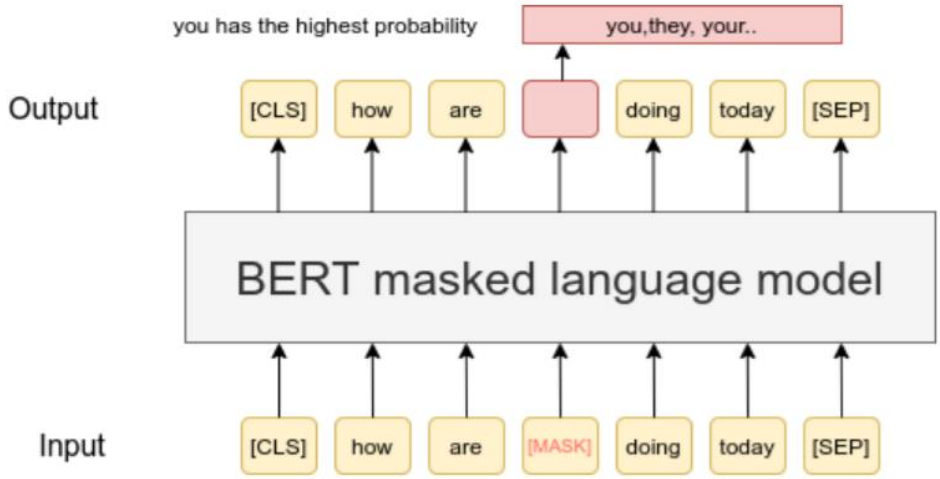

Figura 2: Addestramento Mask Language Model

- **Next Sentence Prediction (NSP):** BERT riceve coppie di frasi come input e deve prevedere se la seconda frase segue la prima nel testo originale. Questo compito aiuta BERT a comprendere la relazione e il flusso del testo, migliorando la sua capacità di catturare la coerenza e il contesto tra le frasi.

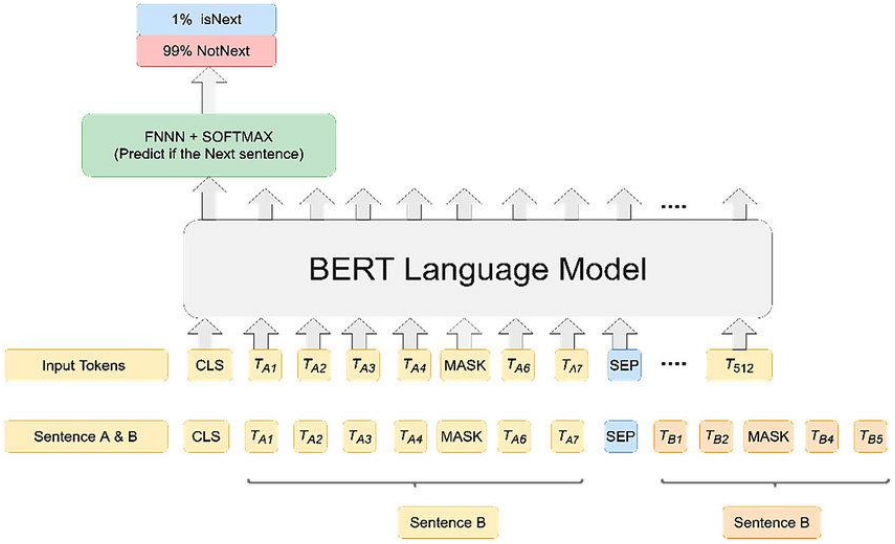

Figura 3: Addestramento Next Sentence Prediction

2. **Fine-tuning**: Dopo il pre-training, BERT viene ulteriormente addestrato su compiti specifici utilizzando dati annotati e supervisionati. In questa fase, il modello viene "sintonizzato" per adattarsi a un compito specifico, come l'analisi del sentiment o il riconoscimento dell'entità nominata. Durante il fine-tuning, i pesi del modello vengono aggiornati utilizzando un insieme di dati annotati per ottimizzare le prestazioni del modello sul compito specifico.

Il pre-training fornisce a BERT una comprensione generale del linguaggio, mentre il fine-tuning permette al modello di specializzarsi su compiti specifici. Questo approccio consente a BERT di essere estremamente flessibile e adattabile a una vasta gamma di applicazioni NLP, garantendo al contempo prestazioni di alto livello su compiti diversi.

Nel nostro progetto eseguiremo solo la parte di fine-tuning, essendo il pre-addestramento estremamente pesante e dispendioso.

2 Dataset utilizzato

Per questo progetto abbiamo il dataset al seguente link.
Il dataset contiene 50.000 recensioni, in lingua inglese, lasciate a diversi film sul sito IMDB. Quando scaricato, il dataset è già diviso in due file di train e test, è stato tuttavia riunito da noi in un unico file e la suddivisione del dataset è stata effettuata successivamente.
Le 50.000 recensioni sono classificate in 'Positive' o 'Negative' se il numero di stelle a loro associato è rispettivamente >=7 o <=4, le recensioni a cui erano associati un numero intermedio di stelle non sono state incluse nel dataset. Le etichette sono state riportate a valori numerici assegnando il valore 0 alle recensioni negative e il valore 1 alle recensioni positive.
Sono presenti 25.000 recensioni positive e 25.000 recensioni negative e non sono presenti più di 30 recensioni lasciate allo stesso film.
È presente anche una parte del dataset non etichettata che contiene recensioni di qualunque punteggio ma, essendo necessarie le etichette sia per il fine tuning che per il calcolo delle metriche, non è stato possibile utilizzare questa parte di dataset.

	text	label
0	Now, I won't deny that when I purchased this o...	0
1	The saddest thing about this "tribute" is that...	0
2	Last night I decided to watch the prequel or s...	0
3	I have to admit that i liked the first half of...	0
4	I was not impressed about this film especially...	0

Figura 4: Prime 5 righe del dataset utilizzato

2.1 Pre-processing del dataset

Il nostro progetto utilizza il modello *BertForSequenceClassification*, che integra il modello pre-addestrato BERT, specificamente il *bert-base-uncased*, caratterizzato da 12 strati transofrmer e 110 milioni di parametri. Questo modello sfrutta la strategia MLM (Masked Language Model). A seguito di BERT, è aggiunto un singolo strato di classificazione lineare che opera su due classi. L'obiettivo principale del modello è classificare una sequenza di testo assegnandole una delle due classi possibili, positiva o negativa, in base al contenuto espresso nella sequenza stessa.
Questo modello richiede che l'input abbia un certo formato nel quale verranno trasformati i nostri dati tramite i passi descritti qui di seguito.

2.1.1 Suddivisione del dataset

Come detto precedentemente, il dataset intero è stato diviso in tre parti, ossia train, validation e test, le cui percentuali di dati sono rispettivamente 70%, 15% e 15% del contenuto dell'intero dataset.
Il dataset di train è quello utilizzato effettivamente per addestrare il modello, il dataset di validation è utilizzato al termine di ogni epoca per verificare l'andamento dell'addestramento, il dataset di test

viene invece utilizzato al termine dell'addestramento per verificare le prestazioni del modello su nuovi dati mai visti prima.
All'interno delle 3 partizioni del dataset è stato mantenuto il rapporto di 50/50 tra recensioni positive e negative.

Dataset iniziale		
Numero di recensioni		50.000
di cui negative		25.000
di cui positive		25.000
Dataset finale		
Train	**Validation**	**Test**
35.000 recensioni	7.500 recensioni	7.500 recensioni
17.500 negative (0)	3.750 negative (0)	3.750 negative (0)
17.500 positive (1)	3.750 positive (1)	3.750 positive (1)

2.1.2 Pre-processing dei dati di input

Il testo delle recensioni viene inizialmente ripulito effettuando i seguenti passi:

1. **Trasformazione in minuscolo**: Uniforma il testo convertendo tutti i caratteri in minuscolo, garantendo coerenza e semplificando le operazioni successive.
2. **Rimozione dei caratteri non alfabetici e della punteggiatura**: Elimina tutti i caratteri non alfabetici e la punteggiatura, lasciando solo parole e segni di interpunzione specifici come ".", "?", "!", ",".
3. **Rimozione degli URL**: Esclude gli URL presenti nel testo, poiché non apportano significato al contesto linguistico.
4. **Rimozione dei tag HTML**: Elimina i tag HTML presenti nel testo, comuni in testi acquisiti da pagine web, per concentrarsi solo sul contenuto testuale.
5. **Trasformazione delle parole in minuscolo**: Tutte le parole vengono convertite in minuscolo per uniformità e per garantire una corretta rimozione delle stopwords, che sono generalmente scritte in minuscolo nei dizionari delle stopwords.
6. **Eliminazione delle stopwords**: Rimuove le stopwords, cioè parole comuni ma poco informative come "il", "che", "di", ecc., che non contribuiscono alla comprensione del significato del testo.
7. **Rimozione degli emoji**: Elimina gli emoji presenti nel testo, che non sono rilevanti per molte analisi linguistiche e potrebbero interferire con la successiva elaborazione.

Successivamente viene effettuata la tokenizzazione delle recensioni, ossia ogni recensione viene divisa in singole parole e ad ognuna di queste viene assegnata una rappresentazione numerica rappresentata da un ID.
Abbiamo poi calcolato la lunghezza massima, in termini di token, delle recensioni presenti nel nostro dataset che è risultata essere di 206.
Viene quindi effettuato il padding delle recensioni che contengono meno token rispetto alla più lunga, ossia vengono aggiunti, al termine dei token della recensione, dei token nulli fino ad arrivare a 206 token; ciò per fare in modo che al modello vengano dati in input tutti dati della stessa lunghezza. Se il modello riceverà dati in input con una lunghezza maggiore di 206 token, troncherà il dato a questa lunghezza.

Ad ogni dato in input viene poi associata una attention mask, ossia un vettore composto da 0 ed 1 in corrispondenza di ogni token che indica al modello dove porre la sua “attenzione”, ossia nelle posizioni in cui è presente un 1 vuol dire che è presente un token, mentre dove è presente uno 0 indica che il token in quella posizione è di padding.
In figura 5 è riportato un esempio di tokenizzazione e attention mask di una parte di una recensione contenuta nel nostro dataset, nella prima colonna sono riportate le parole che compongono la recensione, nella seconda gli ID dei token assegnati alle singole parole, nella terza la attention mask; è possibile notare come cambia da 1 a 0 la attention mask quando i token diventano di padding.

guy	3124	1
preview	19236	1
real	2613	1
film	2143	1
fans	4599	1
lined	7732	1
omaha	12864	1
see	2156	1
great	2307	1
job	3105	1
[SEP]	102	1
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0

Figura 5: Esempio di tokenizzazione di una parte di recensione del nostro dataset

3 Fine tuning del modello

Il modello viene addestrato per 4 epoche, con l'implementazione di uno scheduler di warm-up del learning rate per regolarlo durante le prime fasi dell'addestramento al fine di evitare fluttuazioni e prestazioni deludenti.
È stata creata una lista chiamata *training_stats* per registrare la training loss e la validation loss per ogni epoca, con la *best_eval_loss* utilizzata per tenere traccia della minima validation loss registrata. Durante il ciclo di addestramento, i dati vengono divisi in batch e passati al modello per il calcolo dell'output.
Alla fine di ogni epoca, vengono calcolate la media delle loss di train e di validation, insieme al tempo impiegato per addestrare l'epoca. Inoltre, viene verificato se la validation loss attuale è migliore della precedente *best_eval_loss* e, in tal caso, viene salvato il modello.
Per l'addestramento è stato utilizzato l'ottimizzatore AdamW con i seguenti parametri:

- Learning rate: $2*10^{-5}$
- Epsilon: 10^{-8}

Nella tabella seguente sono riportati i risultati ottenuti dall'addestramento descritto sopra.

Epoca	Training loss	Validation loss	Validation accuracy	Training time
1	0.2971	0.1834	0.9043	0:22:17
2	0.1794	0.1972	0.9030	0:22:21
3	0.0966	0.2114	0.9038	0:22:20
4	0.0504	0.2364	0.9027	0:23:22

Nella figura 6 sono riportati gli andamenti delle due loss rispetto alle epoche di addestramento.

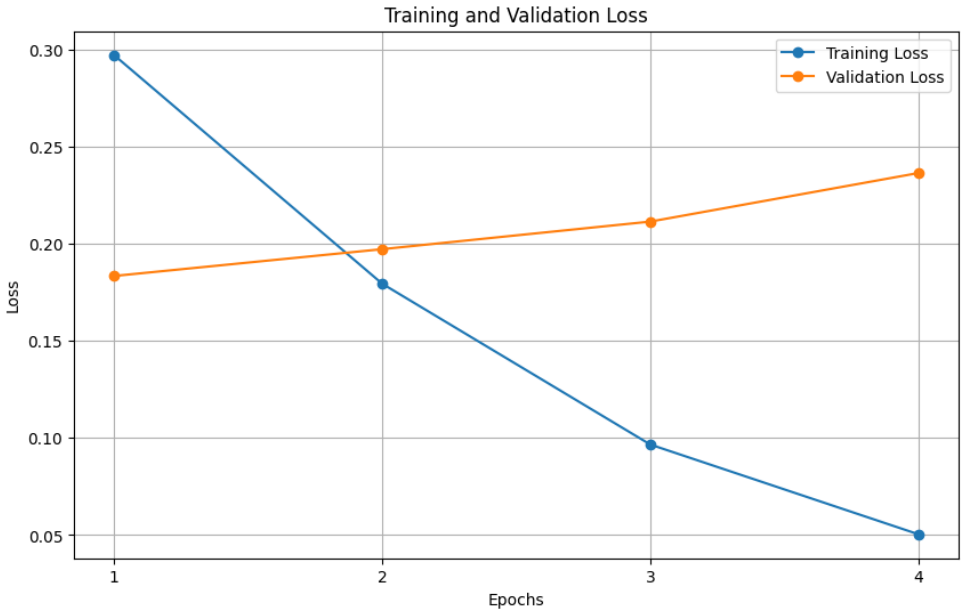

Figura 6: Andamento delle loss di train e test durante l'addestramento

Possiamo notare come la loss di validazione superi quella di train mostrando un chiaro andamento di overfitting. Allora il miglior modello è quello ottenuto dopo la prima epoca di addestramento e andremo ad utilizzare quello per effettuare il test.

4 Test del modello

Le performance del modello ottenuto al termine della prima epoca di addestramento sono state verificate con il dataset di test estratto precedentemente. Per valutarne le prestazioni sono state utilizzate le metriche mostrate in figura 7 con le relative formule per essere calcolate.

$$
\begin{aligned}
precision &= \frac{TP}{TP + FP} \\
recall &= \frac{TP}{TP + FN} \\
F1 &= \frac{2 \times precision \times recall}{precision + recall} \\
accuracy &= \frac{TP + TN}{TP + FN + TN + FP} \\
specificity &= \frac{TN}{TN + FP}
\end{aligned}
$$

Figura 7: Metriche per il test del modello

I risultati ottenuti sono i seguenti:

- **Precision**: 0.8906
- **Recall**: 0.9340
- **F1 score**: 0.9118
- **Accuracy**: 0.9093
- **Specificity**: 0.8845

Nella figura 8 viene riportata la matrice di confusione che indica il numero di previsioni nelle varie categorie (TP, FP, TN ed FN).

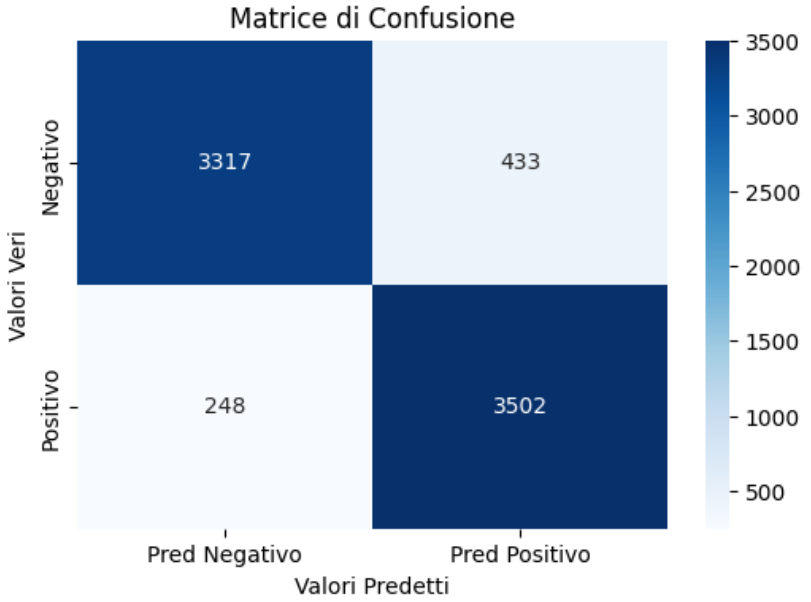

Figura 8: Matrice di confusione dei risultati di test