Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Facoltà di Ingegneria Informatica e dell'Automazione

La sentiment analysis delle recensioni Amazon tramite il framework BERT

Docenti:
Domenico URSINO
Gianluca BONIFAZI

Studenti:
Silvia Ciuffreda
Luca Liberatore
Beatrice Moliterno

ANNO ACCADEMICO 2022/2023

Indice

Elenco delle figure

1 Introduzione

In tale progetto, tramite l'implementazione di un algoritmo di deep learning, verrà effettuato l'addestramento di un modello per il riconoscimento di recensioni positive o negative dei prodotti Amazon. Si utilizzerà Google Colab, una piattaforma che permette di eseguire codice direttamente sul Cloud e che sfrutta Jupyter Notebook.

1.1 L'NLP

Il Natural Language Processing (Figura 1) è un campo di ricerca interdisciplinare che abbraccia informatica, intelligenza artificiale e linguistica, il cui scopo è quello di sviluppare algoritmi in grado di analizzare, rappresentare e quindi "comprendere" il linguaggio naturale, scritto o parlato, in maniera similare o addirittura più performante rispetto agli esseri umani. Ci sono vaste applicazioni di elaborazione del lunguaggio naturale e tra queste è possibile trovare il miglioramento della ricerca, l'analisi e organizzazione di raccolte di documenti di grandi dimensioni, l'analisi dei social media, l'analisi di mercato, l'auotmatizzazione di attività di routine come con l'utilizzo di chatbot o assistenti digitali.

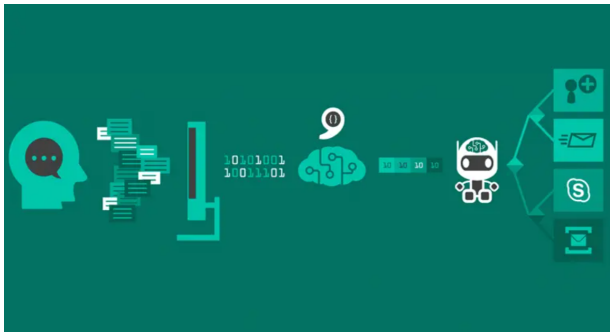

Figura 1: L'NLP

Negli ultimi anni si è assistito alla nascita di nuovi approcci, che integrano l'elaborazione del linguaggio naturale con gli algoritmi deep learning, producendo risultati straordinari in differenti scenari applicativi.

1.2 BERT

BERT (*Bidirectional Encoder Representations from Transformers*) è un framework open source di machine learning per l'NLP; in particolare, Bert si basa sull'architettura del *transformer*, rilasciata da Google nel 2017. Nella sua forma più pura, un transformer è costituito da due meccanismi distinti: un codificatore e un decodificatore; il primo effettua una codifica del testo ricevuto in input, mentre il secondo genera una decodifica della previsione fatta in base all'attività per cui è stato addestrato il modello. A differenza dei modelli direzionali, che leggono l'input di testo in sequenza (da sinistra a destra o da destra a sinistra), il codificatore del Trasformer legge l'intera sequenza di parole contemporaneamente; dunque è considerato *bidirezionale*. Questa sua caratteristica permette al modello di apprendere il contesto di una parola in base a tutto ciò che la circonda, non solo basandosi sulla parola precedente e su quella successiva.

Poiché i trasformes possono elaborare i dati in qualsiasi ordine, essi consentono l'addestramento su quantità di dati maggiori di quanto fosse possibile prima della loro esistenza. Questo, a sua volta, ha facilitato la creazione di modelli pre-addestrati come BERT, che è stato addestrato su enormi quantità di dati linguistici prima del suo rilascio.

Durante le fasi di training, il modello BERT riceve in input una sequenza di frasi, sotto forma di sequenze di token, e restituisce in output una sequenza di vettori di una determinata dimensione e ogni vettore corrisponde a un token di input con lo stesso indice. Per addestrare il modello linguistico Bert ricorre a due possibili strategie:

- Masked Learning Model (Figura 2), in cui vengono mascherate casualmente alcune parole di una frase durante l'addestramento e il modello cerca di predire le parole mancanti in base al contesto circostante.

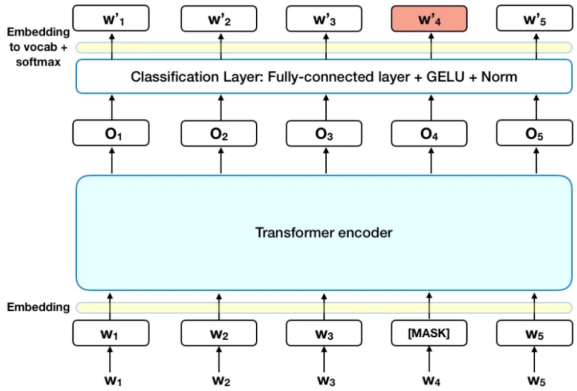

Figura 2: Lo schema di funzionamento del Masked Learning Model

- Next Sentence Prediction (Figura 3), il quale coinvolge la comprensione delle relazioni tra due frasi consecutive e prevede se tale coppia è logicamente legata ed è sequenziale, oppure se la loro correlazione è casuale.

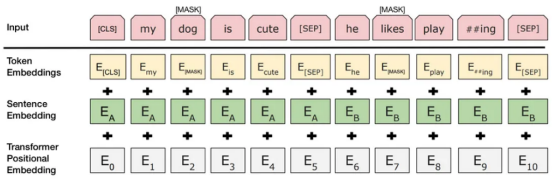

Figura 3: Lo schema di funzionamento del Next Sentence Prediction

2 Il dataset

Il dataset preso in considerazione nel suddetto progetto è stato creato per il paper 'From Group to Individual Labels using Deep Features', Kotzias et. al,. KDD 2015 ed reperibile al seguente link:

https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

In questo link è possibile reperire tre file in formato .txt contenenti frasi etichettate con sentimento positivo o negativo, estratte, rispettivamente, da recensioni di prodotti dal sito Amazon.com, recensioni di film dal sito imdb.com e recensioni di ristoranti dal sito yelp.com. Si è scelto di analizzare il file txt corrispondente alle recesioni dei prodotti amazon.

2.1 La struttura del dataset

Questo dataset è costituito da 1000 righe e 2 colonne; le righe contengono 500 frasi etichettate con sentimento positivo e 500 frasi etichettate con sentimento negativo. In particolare, come è mostrato in Figura 4, la prima colonna del dataset, definita dal campo *text*, comprende tutte le proposizioni, mentre la seconda colonna, definita dal campo *label*, deinisce le etichette "0" se il sentimento associato alla frase è negativo, "1" se il sentimento associato alla frase è positivo.

	text	label
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
3	Tied to charger for conversations lasting more...	0
4	The mic is great.	1
5	I have to jiggle the plug to get it to line up...	0
6	If you have several dozen or several hundred c...	0
7	If you are Razr owner...you must have this!	1
8	Needless to say, I wasted my money.	0
9	What a waste of money and time!.	0

Figura 4: Le prime 10 righe del dataset

2.2 Le prime elaborazioni del dataset

Una fase da non dimenticare prima di addestrare il modello è l'eliminazione delle parole chiave, necessaria per rendere i dati analizzabili dal modello BERT.

Il dataset in questione raccoglie frasi che potrebbero contenere errori di battitura, simboli particolari o emoticon e questo potrebbe scaturire difficoltà da parte dell'encoding di BERT. Dunque, è stato importante eliminare tutti i caratteri che avrebbero portato ad una mal interpretazione e che non sarebbero stati utili per la comprensione del testo.

3 Il pre processamento e l'addestramento del modello

Il modello utilizzato in tale progetto è il *BertForSequenceClassification*, il quale combina il modello preallenato BERT, il *bert-base-uncased*, composto da 12 layer e 110 milioni di parametri, che sfrutta la strategia MLM, e, a valle di quest'ultimo, un singlo strato di classificazione lineare a due classi. L'obiettivo di tale modello è quello di assegnare una classe ad una determinata sequenza di testo e, in tal caso, l'obiettivo sarà quello di assegnare un sentimento positivo o negativo alle proposizioni raccolte nel dataset.

3.1 La divisione del dataset in train e test

Il dataset è stato diviso in:

- *train set*, che comprende l'80% del dataset, utilizzato per addestrare il modello; contiene i dati che il modello userà per apprendere le relazioni utili alla predizione.
- *test set*, che comprende il restante 20% del dataset, utilizzato per valutare le prestazioni del modello dopo l'addestramento.

3.2 Il pre processamento dell'input

Un'operazione importante da svolgere è la *tokenizzazione* sul train set, la quale trasforma le frasi in un insieme di token, per poi essere salvate in array che costituiscono le matrici chiamate *tensori*; le proposizioni trasformate devono avere tutte la stessa lunghezza e ciasuna non deve superare i 512 token. Tutto questo è dovuto alle caratteristiche di elaborazione di Bert. Il *tokenizer* utilizzato per tale operazione è il *BertTokenizer* che si basa sul modello *bert-base-uncased*.

E' importante capire quale è la lunghezza massima delle proposizioni trasformate in token e, dal codice elaborato per la sua ricerca, si evince che il valore è pari a 26, come illustrato in Figura 5.

```
Max sentence length:  26
```

Figura 5: Il valore di lunghezza massima di una frase

Dunque, si è scelto di fissare la lunghezza di ciasun tensore uguale a 26; se l'input risulta essere più corto della lunghezza massima, si effettua l'operazione di *padding*, che aggiunge "zero" per raggiungere la dimensione scelta; inoltre, viene definita l'*Attenction Mask*, che evidenzia dove si concentra l'informazione effettiva, assegnando "1" al token con informazione utile e "0" al token senza informzione. Per ogni frase è stata, quindi, effettuata la tokenizzazione.

In Figura 6 è riportata una tabella in cui si evidenzia come è stata effettuata la tokenizzazione di una frase casuale e la sua l'Attenction Mask; si nota che per ottenere un tensore di dimensione 26, sono stati aggiunti 20 "zero" di padding che non danno alcuna informazione tile

worst phone ever months

Tokens	Token IDs	Attention Mask
[CLS]	101	1
worst	5409	1
phone	3042	1
ever	2412	1
months	2706	1
[SEP]	102	1
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0
[PAD]	0	0

Figura 6: Un esempio di tokenizzazione

Una volta effettuata la tokenizzazione i dati sono pronti per essere dati in pasto al modello.

3.3 Divisione del dataset in train e validation

Una volta effettuato il pre processamento dei dati del train set, diventato un dataset di tensori, è stato importante separare i dati in train set e validation set, utilizzato per valutare le prestazioni del modello durante l'addestramento e per testare i suoi iperparametri. Dunque, l'80% di tutti gli elementi sottoposti ad encoding, descritto nel paragrafo precedente, è stato inserito nel train set e il restante 20% nel validation set.

3.4 L'addestramento del modello

L'addestramento del modello è stato effettuato considerando 2 *epoche* e *batch size* pari a 16. L'ottimizzatore utilizzato è l'*Adam* e gli iperparametri scelti sono:

- *learning rate*: $5 \cdot e^{-5}$
- *epsilon*: $1 \cdot 10^{8}$

Successivamente, sono stati valutati gli andamenti della *training loss*, *validation loss* e *validation accuracy*, mostrati sia in Figura 7.

epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.58	0.41	0.84	0:03:21	0:00:13
2	0.35	0.39	0.85	0:03:21	0:00:13

Figura 7: L'andamento dell'addestramento

In Figura 8 è riportato un grafico per confrontare i valori di *training loss* e *validation loss*, già evidenziati nella tabella.

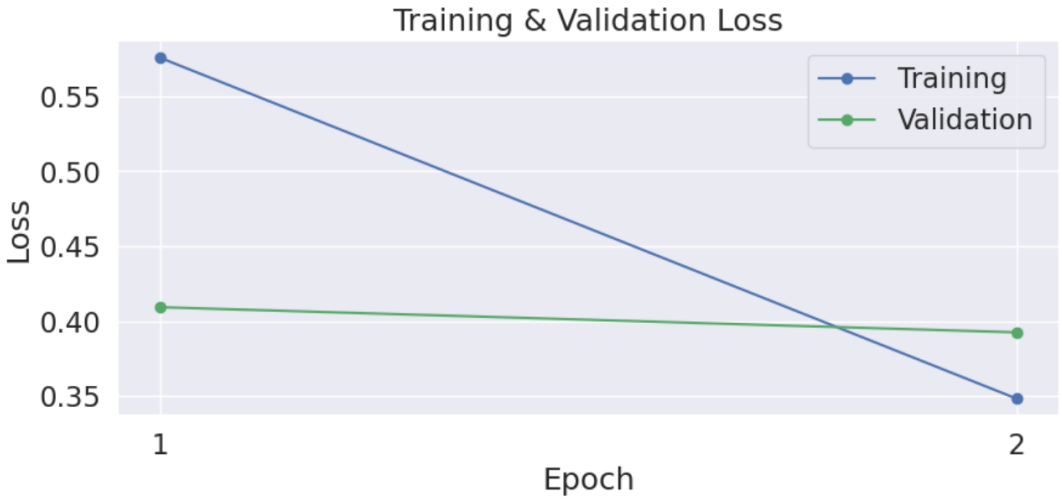

Figura 8: Il confronto tra training e validation loss

Si nota che la loss del training decresce notevolmente, a differenza della loss della validation che diminuisce, ma, leggermente. Si evince un problema di overfitting già all'epoca 2, in quanto la *validation loss* risulta maggiore della *training loss*.

4 I risultati ottenuti nella fase di test

E' stato effettuato, per il test set, lo stesso pre processamento eseguito per i dati di train e validation, descritto nel paragrafo 3.2. In seguito, è stata eseguita la predizione sui dati modificati e valutato la bontà del modello, utilizzando alcune metriche e la *confusion matrix*.

4.1 Le metriche per la valutazione del modello

Le metriche utilizzate per la valutazione delle performance del test sono l'*accuracy*, la *precision*, l'*f1_ score* e la *recall*. Queste sono state calcolate tramite la libreria python "scikit learn" e la descrizione di ciascuna metrica è riassunta nella tabella in Figura 9.

Metrica	Descrizione
Accuracy	$\frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative}$
F1 score	$\frac{2 * (Precision * Recall)}{(Precision + Recall)}$
Precision	$\frac{TruePositive}{TruePositive + FalsePositive}$
Recall	$\frac{TruePositive}{TruePositive + FalseNegative}$

Figura 9: Le metriche utilizzate

I risultati ottenuti sono i seguenti:

- *Accuracy*: 0.86
- *f1_ score*: 0.85
- *Precision*: 0.88
- *Recall*: 0.82

4.2 La confusion matrix

La confusion matrix è uno strumento che permette di valutare quanti errori ha commesso il modello. In questo progetto si lavora con un classificatore binario e, per tale ragione, si avrà una matrice 2X2. Le righe indicano le classi effettive, mentre le colonne indicano le classi di previsione.

La confusion matrix (non normalizzata) del modello è riportata in Figura 10.

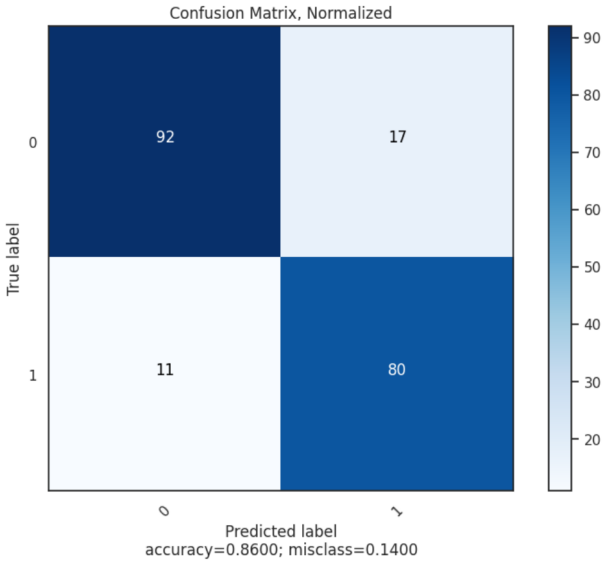

Figura 10: Le confusion matrix

Sulla diagonale principale è possibile visualizzare il valore totale degli elementi classificati correttamente, ovvero 172, di cui 92 corrispondono ai "true positive", mentre, 80 corrispondono ai "true negative".