UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione

Sentiment analysis di recensioni di videogiochi di Steam tramite Google BERT

Progetto Data Science

Docenti:
Prof. Ursino Domenico
Dott. Marchetti Michele

Studenti:
Cardoni Lorenzo
El Mechri Rahmi
Novelli Giovanni

Anno Accademico 2023-2024

Indice

Elenco delle figure

1 Introduzione

In questo progetto, si intende implementare un modello di deep learning per addestrare un sistema di sentiment analysis, che sarà in grado di riconoscere automaticamente recensioni positive o negative sui videogiochi presenti nel catalogo di Steam. Per realizzare l'addestramento e l'analisi, si utilizzerà Kaggle, una piattaforma basata sul cloud che consente di eseguire codice tramite un ambiente Jupyter Notebook, offrendo risorse computazionali avanzate per supportare il processo di training del modello.

1.1 L'NLP

Il Natural Language Processing (NLP) è un campo di ricerca interdisciplinare che unisce informatica, intelligenza artificiale e linguistica, con l'obiettivo di sviluppare algoritmi capaci di analizzare, rappresentare e "comprendere" il linguaggio naturale, sia scritto che parlato, in modo simile o addirittura più accurato rispetto agli esseri umani. Le applicazioni del NLP sono molto ampie e comprendono il miglioramento delle capacità di ricerca, l'analisi e l'organizzazione di grandi collezioni di documenti, l'analisi dei social media, lo studio di mercato e l'automatizzazione di attività di routine, come avviene con chatbot e assistenti digitali. Alcune delle sue applicazioni sono mostrate in Figura 1.1

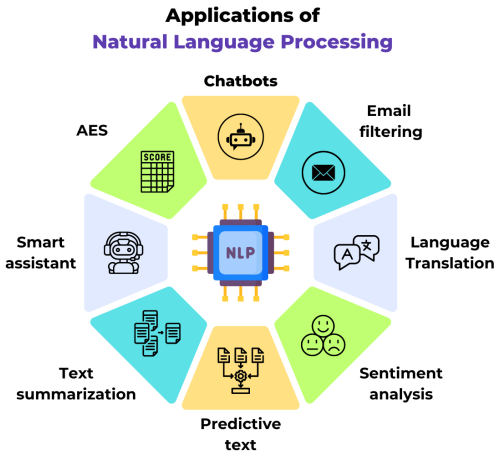

Figura 1.1: NLP

Negli ultimi anni, sono emersi nuovi approcci che combinano l'elaborazione del linguaggio naturale con algoritmi di deep learning, portando a risultati straordinari in diversi contesti applicativi. Queste tecniche avanzate hanno permesso progressi significativi, migliorando

l'accuratezza e le capacità dei modelli NLP in molteplici scenari, dall'analisi testuale alla generazione automatica di contenuti.

1.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) è un framework di machine learning open source sviluppato per affrontare sfide di NLP, basato sull'architettura del transformer introdotta da Google nel 2017. L'architettura di un transformer è suddivisa in due componenti principali: il codificatore e il decodificatore.
Il codificatore processa e rappresenta il testo fornito in input, mentre il decodificatore produce un'interpretazione della previsione a seconda del compito specifico per cui il modello è stato addestrato. Diversamente dai modelli sequenziali, che elaborano il testo parola per parola in una sola direzione (da sinistra a destra o viceversa), il codificatore del transformer legge l'intera sequenza di parole contemporaneamente, risultando quindi bidirezionale.
Questo approccio permette a BERT di comprendere una parola nel contesto globale della frase, valutando tutte le parole circostanti piuttosto che limitarsi a quelle immediatamente vicine.
Essendo progettati per gestire dati in modo non sequenziale, i transformer facilitano l'addestramento su ampi volumi di dati, superando i limiti dei modelli precedenti. Questo ha reso possibile la creazione di modelli pre-addestrati come BERT, allenati su enormi corpus linguistici prima di essere rilasciati. Durante il training, BERT riceve una sequenza di frasi sotto forma di token e restituisce una sequenza di vettori di dimensioni predefinite, dove ogni vettore rappresenta uno specifico token in input.
Per addestrare il modello linguistico BERT, vengono impiegate due strategie principali:

- Masked Language Model: durante il training, alcune parole di una frase vengono mascherate in modo casuale. Il modello è quindi addestrato a predire le parole nascoste basandosi sul contesto circostante, migliorando così la sua capacità di comprendere il significato globale della frase, come mostrato in Figura 1.2.

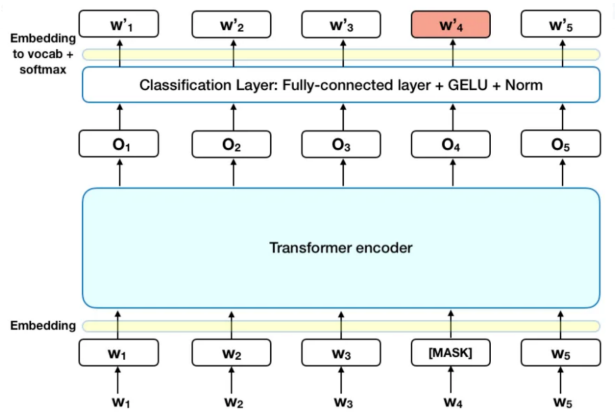

Figura 1.2: Masked Language Model

- Next Sentence Prediction: questa strategia aiuta il modello a comprendere le relazioni tra frasi consecutive. Viene infatti presentata al modello una coppia di frasi, e l'obiet-

tivo è predire se la seconda frase è logicamente collegata alla prima o se sono frasi non correlate, come mostrato in Figura 1.3.

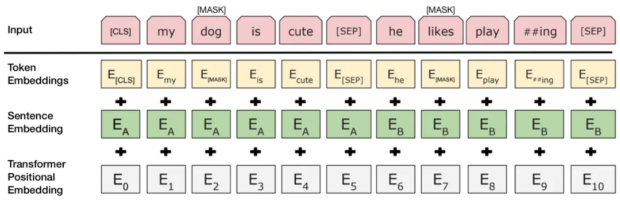

Figura 1.3: Next Sentence Prediction

Ricapitolando, questi due metodi permettono a BERT di apprendere sia il contesto locale delle parole all'interno di una frase, sia il contesto più ampio tra frasi consecutive, rendendolo particolarmente efficace per una vasta gamma di applicazioni di NLP.

2 Dataset

Il dataset scelto per questo progetto è composto da oltre 6,4 milioni di recensioni pubblicamente disponibili in inglese, provenienti dalla sezione Steam Reviews del negozio Steam gestito da Valve.
Il dataset utilizzato per questa analisi è stato ottenuto dal sito Kaggle (link).
Questa vasta raccolta di recensioni offre un'opportunità unica per analizzare i fattori che influenzano la soddisfazione e l'insoddisfazione degli utenti riguardo ai giochi e ai generi, nonché per esaminare l'evoluzione del sentiment nel tempo.

2.1 La struttura del dataset

La struttura del dataset scelto è rappresentata in Figura 2.1

	app_id	app_name	review_text	review_score	review_votes
0	10	Counter-Strike	Ruined my life.	1	0
1	10	Counter-Strike	This will be more of a "my experience with th...	1	1
3	10	Counter-Strike	• Do you like original games? • Do you like ga...	1	0
4	10	Counter-Strike	Easy to learn, hard to master.	1	1
5	10	Counter-Strike	No r8 revolver, 10/10 will play again.	1	1
...	...	...	...	...	...
6417101	99910	Puzzle Pirates	I really ove this game but it needs somethings...	-1	0
6417102	99910	Puzzle Pirates	Used to play Puzzel Pirates 'way back when', b...	-1	0
6417103	99910	Puzzle Pirates	This game was aright, though a bit annoying. W...	-1	0
6417104	99910	Puzzle Pirates	I had a nice review to recommend this game, bu...	-1	0
6417105	99910	Puzzle Pirates	The puzzles in this game are fun, but you have...	-1	0

Figura 2.1: Struttura Dataset

Il dataset utilizzato per questo progetto contiene circa 6,4 milioni di righe, ciascuna rappresentante una recensione di un gioco. Ogni recensione è descritta da cinque colonne:

- l'ID del gioco a cui la recensione è associata
- il nome del gioco
- il testo della recensione
- "review_score" rappresenta il sentimento della recensione, indicando se la recensione consiglia o meno il gioco

- "review_score" (il voto della recensione), cioè se la recensione è stata consigliata da un altro utente oppure no

Nel dataset, la colonna **review_score** rappresenta il sentimento della recensione, indicando se la recensione consiglia o meno il gioco. Le recensioni con sentiment positivo sono contrassegnate con il valore "1", mentre quelle con sentiment negativo sono contrassegnate con il valore "-1.". La colonna **review_votes** indica l'utilità della recensione: un valore di "1" significa che la recensione è stata consigliata da altri utenti (indicando che è ritenuta utile), mentre un valore di "0" indica che la recensione non è stata consigliata.

2.2 Preprocessing del dataset

2.2.1 Selezione delle features e trasformazione

Come descritto precedentemente in 2.1 il dataset contiene più colonne di quelle necessarie. Per questo motivo abbiamo selezionato unicamente le due colonne necessarie ai fini della classificazione, ovvero quella contenente il testo della recensione e quella del sentimento della recensione (**review_score**), la quale indica se la recensione consiglia o meno il gioco.
Abbiamo quindi cambiato il nome delle colonne respettivamente in **'text'** e **'label'**.
A questo punto abbiamo applicato delle trasformazioni per rendere consono il formato del dataset.
La prima trasformazione eseguita è l'eliminazione delle righe contenenti valori nulli.
La seconda trasformazione riguarda invece la colonna 'label'. In questa colonna le recensioni con sentiment positivo presentano valore 1 mentre quelle con sentiment negativo il valore -1. Abbiamo modificato il valore delle recensioni con sentiment negativo a 0.
A questo punto, poichè il dataset presenta oltre 6 milioni di righe, che risulterebbero eccessivamente onerose per un training, abbiamo deciso di selezionarne un numero adeguato, ovvero 50 mila recesioni per tipologia (positive e negative), ottenendo in totale 100 mila recensioni.

2.2.2 Suddivisione del dataset in train e test

Per le operazione di training, validation e testing abbiamo deciso di applicare uno split del dataset associando rispettivamente il 70%, 15% e 15% dei dati alle 3 fasi.
Abbiamo mantenuto all'interno delle 3 partizioni il rapporto iniziale, affinchè il dataset rimangia bilanciato, così da evitare sbilanciamenti in fase di classificazione.
In tabella 2.1 sono esplicitati il numero di righe per partizione.

Dataset	Righe totali	Sentiment Positivo	Sentiment Negativo
Iniziale	100000	50000	50000
Training	70000	35000	35000
Validation	15000	7500	7500
Testing	15000	7500	7500

Tabella 2.1: Suddivisione del dataset

3 Fine tuning

Nella fase di fine tuning abbiamo effettuato il training del modello affinchè BERT si adatti al nostro task. Abbiamo addestrato il modello per 4 epoche, nonostante il numero di epoche consigliato in questi casi sia 2, poichè fini di ricerca ci interessa studiare l'andamento del comportamento del modello.

Abbiamo utilizzato l'ottimizzatore AdamW per l'addestramento, utilizzando i parametri consigliati per questo tipo di training ovvero:

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

Alla fine di ogni epoca si calcolano train loss e validation loss, che servono a quantificare il comportamento del modello. In questa fase si tiene traccia del migliore valore di validation loss ottenuto, e si salva il modello solamente quando si ottengono risultati migliori.

In tabella 3.1 ed in figura 3.1 mostriamo l'andamento di training loss e validation loss durante l'addestramento. È facilmente verificabile dal pattern presente che, come previsto, il modello presenti un andamento di overfitting a partire dalla terza epoca, pertanto il modello salvato ed utilizzato per il testing è quello ottenuto al termine della seconda epoca.

Epoca	Training loss	Validation loss	Validation Accuracy	Training time
1	0.4379 ●	0.3907 ●	0.8027 ●	30:42
2	0.3456 ●	0.3889 ●	0.8050 ●	30:53
3	0.2668 ●	0.4237 ●	0.8094 ●	30:55
4	0.2076 ●	0.4794 ●	0.8080 ●	30:57

Tabella 3.1: Andamento addestramento

Figura 3.1: Andamento Train e Validation loss

4 Testing

Riportiamo in questa sezione dell'elaborato le metriche ottenute in fase di testing del modello. Ricordiamo che il modello utilizzato è quello ottenuto alla fine della seconda epoca, poichè presenta la miglior validation loss. Sono state utilizzate le metriche più significative, presentate in tabella 4.1. In figura 4.1 è invece possibile vedere la matrice di confusione ottenuta in fase di testing.

Metrica	**Valore**	**Formula**
Accuracy	0.8000	$\frac{TP+TN}{TP+FN+TN+FP}$
Precision	0.8641	$\frac{TP}{TP+FP}$
Recall	0.7180	$\frac{TP}{TP+FN}$
F1 Score	0.7843	$\frac{2\times precision\times recall}{precision+recall}$
Specificity	0.8838	$\frac{TN}{TN+FP}$

Tabella 4.1: Metriche del modello

Come possiamo vedere, il modello presenta un buon comportamento, ma non eccellente, attestandosi su un'accuracy dell'80%. Questo è probabilmente dovuto ai termini e ai modi di dire utilizzati dalla community di videogiocatori, che spesso non sono presenti nel vocabolario inglese standard.

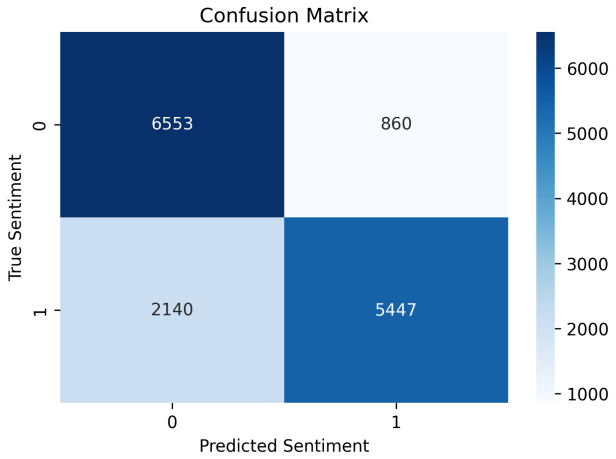

Figura 4.1: Matrice di confusione