UNIVERSITÀ
DELLA CALABRIA

Dipartimento di ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES)
Corso di Laurea Magistrale in Ingegneria Informatica

Relazione del PROGETTO DI
ANALISI DI SOCIAL NETWORK E MEDIA

SENTIMENT ANALYSIS E NLP SUI TESTI DEI POST DELLA PIATTAFORMA SOCIAL: TIKTOK

Studente:
Mattia Gatto, 216649

Docente:
Andrea Tagarelli

Indice

1. Introduzione

Come progetto per il corso di Analisi di Social Network e media è stato preso in considerazione il nuovo social network emergente, TikTok, conosciuto anche come Douyin in Cina. TikTok è un social network cinese lanciato nel settembre 2016, inizialmente chiamato col nome musical.ly. Attraverso l'app realizzata per la piattaforma, gli utenti registrati possono creare brevi clip musicali di durata variabile (da 15 a 180 secondi) ed eventualmente modificare la velocità di riproduzione, aggiungere filtri ed effetti particolari ai loro video. In Cina l'applicazione non è la stessa pubblicata in Occidente ed è più sviluppata, integrando anche funzioni per l'Internet marketing.

TikTok è disponibile in oltre 150 paesi, ha oltre 1 miliardo di utenti ed è stato scaricato oltre 200 milioni di volte solo negli Stati Uniti. Il pubblico di destinazione include persone di età compresa tra 13 e 60 anni, e ha ufficialmente oltre 1 miliardo di utenti attivi mensilmente.

Il progetto si pone l'obbiettivo di prendere attraverso delle API più informazioni possibili dalla piattaforma e successivamente effettuare Sentiment Analysis e NLP sui testi dei post pubblicati dagli utenti, tenendo fortemente in considerazione l'influenza del sentiment delle emoji nelle intestazioni a tali post.

2. Estrazione dei dati

Per estrapolare i dati dalla piattaforma, attraverso varie ricerche nel web, non sono state trovate API ufficiali, ma solo librerie di terze parti incluse in Python.

Per prendere dati e informazioni dai vari utenti nella piattaforma social, è stato utilizzato TiktokApi, un wrapper API non ufficiale per "TikTok.com" in Python. Con questa API è possibile chiamare la maggior parte delle tendenze e recuperare informazioni specifiche sull'utente e molto altro ancora. A causa degli aggiornamenti sui certificati della piattaforma utilizzando le funzioni messe a disposizione dalla libreria, si riscuotono numerosi errori.

Da queste osservazioni, è stato dedotto che è possibile effettuare solamente richieste "getUser(user_id)", per ricavare non solo le informazioni sull'user (i "Like" totali, ecc.), ma anche per prendere tutte le informazioni (i link, musica utilizzata, effetti, ecc.) relative agli ultimi sei post pubblicati.

Ecco qui il link github della libreria usata: https://github.com/davidteather/TikTok-Api

Per ricavare le migliori informazioni, attraverso la funzione menzionata sopra, sono stati presi in considerazione gli utenti appartenenti alla top_1000 in termini di popolarità sulla piattaforma TikTok, prendendo gli "user_id" dalla sito esposto nel seguente link: https://hypeauditor.com/top-instagram/?p=1 .

Questi utenti inseriti successivamente in una lista Python (nella precisione sono stati inseriti 711 user), sono stati utilizzati per ricavare l'insieme dei dati, relativi ai sei ultimi post pubblicati da questi ultimi.

2.1. Preparazione dei dati

Il passo successivo è stato quello di preparare nella maniera più opportuna i dati, infatti, in una prima fase sono state aggregate in un unico dataset l'insieme delle informazioni relative ai 711 utenti, sono state inserite in formato tabellare dove la singola riga, ha varie colonne relative alle informazioni riguardanti:

- l'utente
- il singolo post pubblicato.

In seguito, per ogni testo sono state estrapolate le emoticon, le quali non sempre risultanti in una intestazione di un post, sono state codificate nella loro versione semantica testuale, in modo da successivamente poter processare la colonna dei testi ed eliminare le emoticon, per poter inserire al loro posto l'informazione dell'emoji calcolata, sotto forma di parole.

Per le colonne interessate si è preferito utilizzare una libreria chiamata GoogleTranslator per portare tutte le frasi in una unica lingua comune, ossia l'inglese.

Infine, in una ultima fase è stato definito un approccio di Language Processing diviso in cinque fasi per ridurre la complessità delle singole frasi:

1. Tokenizatione:
   Definizione della frase divisa per token.

2. Rimozione della punteggiatura:
   Non utile per l'indicizzazione.

3. Rimozione delle word nella StopWords list:
   La StopWords list rappresenta l'insieme delle parole più usate per connettere frasi all'interno di una lingua, nel nostro caso l'inglese.

4. Stemming:
   Tecnica usata per portare parole simili in un format comune.

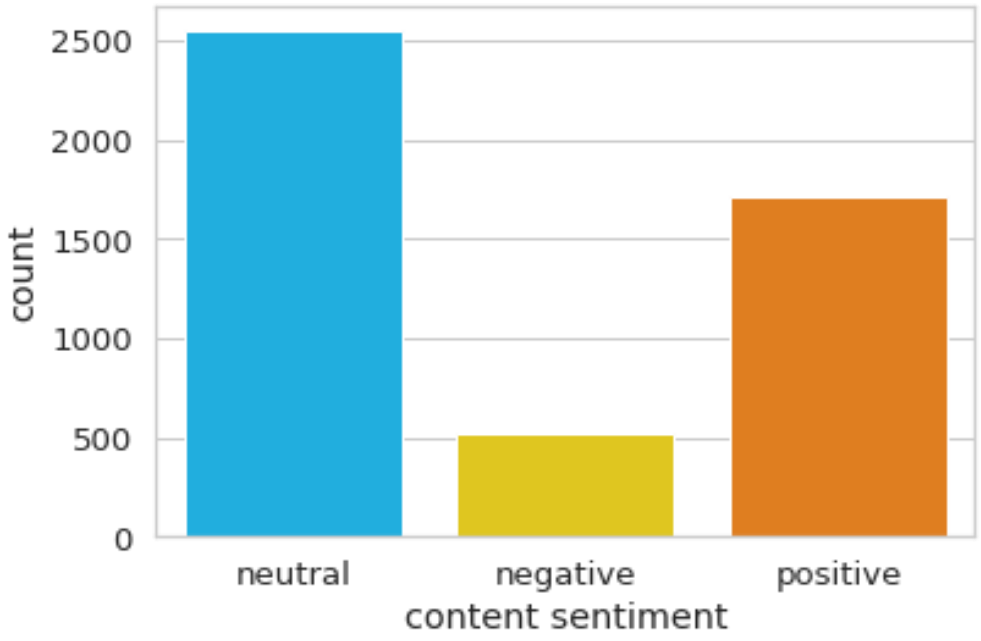
5. Lemmatization:
   Tecnica per trasformare i verbi nella loro forma base.

6. Sostituzione nella frase originale, con la nuova frase elaborata.

Calcoliamo infine il sentiment di ogni frase attraverso la libreria nltk, sfruttando il metodo "polarity_scores(text)" per estrapolare da ogni frase il sentiment e quindi definire una nuova colonna in cui se la frase ha sentiment pari a :

- 0 = neutral
- 1 = negative
- 2 = positive

Ecco una rappresentazione grafica delle occorrenze del sentiment:



3. NLP

Per il processo di Natural Language è stato usato l'algoritmo RoBERTa, ma prima di fare partire il training è stato prodotto un processamento ulteriore dei dati che BERT e ogni sua ottimizzazione purtroppo richiede.

3.1. Descrizione del Modello

L'insieme delle rappresentazioni dell'encoder bidirezionale di Transformers, o BERT , rappresenta una tecnica rivoluzionaria di pre-addestramento auto-supervisionato che impara a prevedere sezioni di testo intenzionalmente nascoste (mascherate). Fondamentalmente, è stato dimostrato che le rappresentazioni apprese da BERT si generalizzano bene alle attività a valle. Quando BERT venne rilasciato per la prima volta nel 2018 ottenne fin da subito risultati all'avanguardia su molti set di dati di riferimento a NLP.

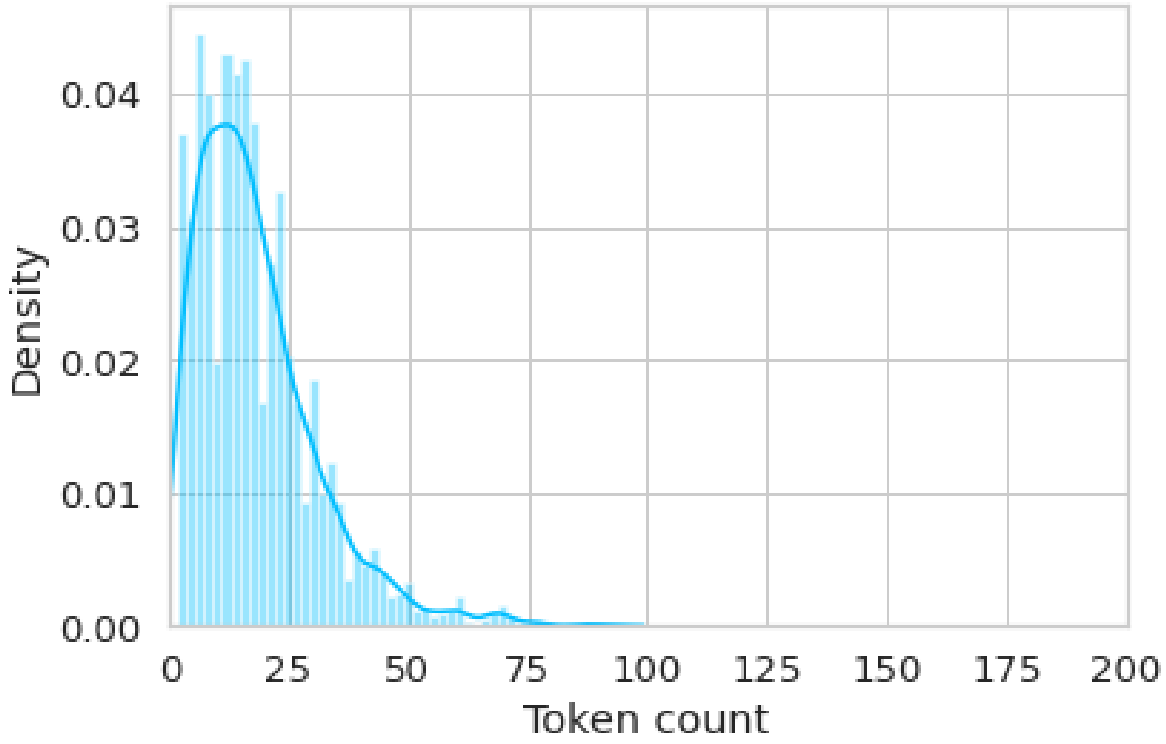
RoBERTa si basa sulla strategia di mascheramento linguistico di BERT e modifica i suoi iperparametri chiave, inclusa la rimozione dell'obiettivo di preformazione della frase successiva di BERT e la formazione con mini-batch e tassi di apprendimento molto più grandi. RoBERTa è stato anche addestrato su un ordine di grandezza di dati in più rispetto a BERT, per un periodo di tempo molto più lungo. Ciò ha consentito alle rappresentazioni RoBERTa di generalizzare ancora meglio le attività a valle rispetto a BERT. Ed è proprio per questa sua efficacia in più risetto al classico BERT che è stato preferito utilizzare l'algoritmo RoBERTa.

3.2. Preparazione, training e valutazione

In questa fase è stato definito il tokenizzatore pre-addestrato "RobertaTokenizerFast" per definire i vari token all'interno delle frasi testuali nella colonna del dataset ottenuto.

Il valore relativo al "max_length" da dare in input al tokenizer insieme al text ovviamente, è stato definito attraverso questa semplice analisi grafica, dalla quale si evince come il valore di lunghezza delle frasi sia quasi sempre al di sotto di un valore pari a 75, e quindi per questo e per una maggior precisione, è stato inserito un valore leggermente più elevato ossia 100.

Ecco qui la rappresentazione grafica:



In seguito, si sono costruiti rispettivamente due strutture dati contenenti ognuna di esse label e text da valutare successivamente usando il modello, di cui la prima verrà presa per il traning del modello, e la seconda per una sua valutazione.

Utilizzando RoBERTa per la struttura di dati contenente frasi testuali, con labels pari al sentiment di tali frasi sono stati ottenuti buoni risultati.

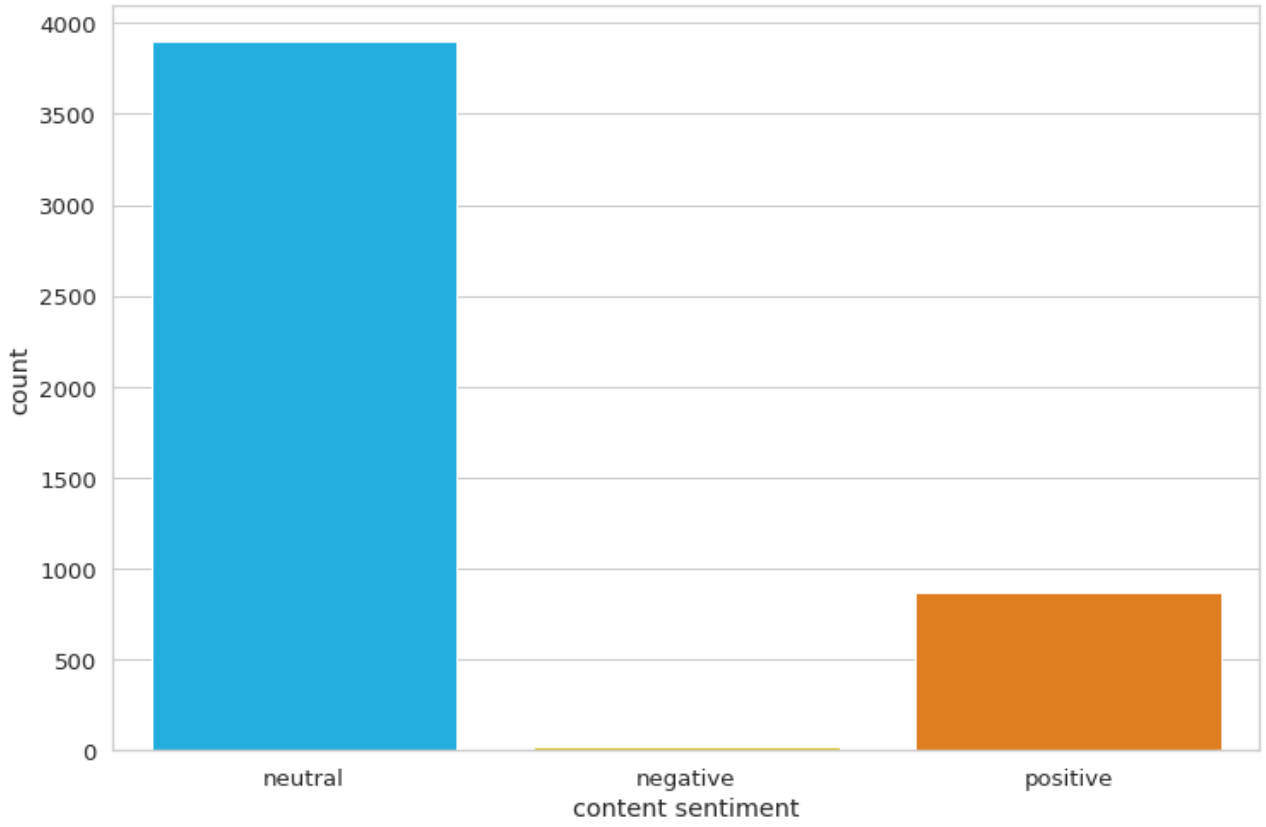
Ecco una rappresentazione del classification_report ottenuto dalla valutazione del modello addestrato, sul validation_set:

	precision	recall	f1-score	support
neutral	0.99	0.99	0.99	521
negative	0.97	0.93	0.95	105
positive	0.97	0.99	0.98	331
accuracy			0.98	957
macro avg	0.98	0.97	0.97	957
weighted avg	0.98	0.98	0.98	957

3.3. Ulteriore analisi di NLP

Come ulteriore processamento dei dati, sono state prese questa volta le emoji relative alla singola frase, e attraverso una libreria chiamata “emosent-py” è stato ricavato il sentiment medio delle emoticon per ogni frase, e il tutto è stato inserito in una colonna del nostro dataset.

Ecco una rappresentazione grafica delle occorrenze del sentiment, basato esclusivamente sulle emoticon:



Utilizzando le identiche procedure di tokenizzazione e costruzione delle strutture dati per il training del modello e il validation, ma questa volta considerando come lista di assegnamento delle labels la nuova colonna creata per il sentiment delle emoticon, abbiamo ottenuto un modello che attraverso una valutazione finale, in termini di classification_report prodotto dall’applicazione del modello al validation_set, ha prodotto i seguenti risultati:

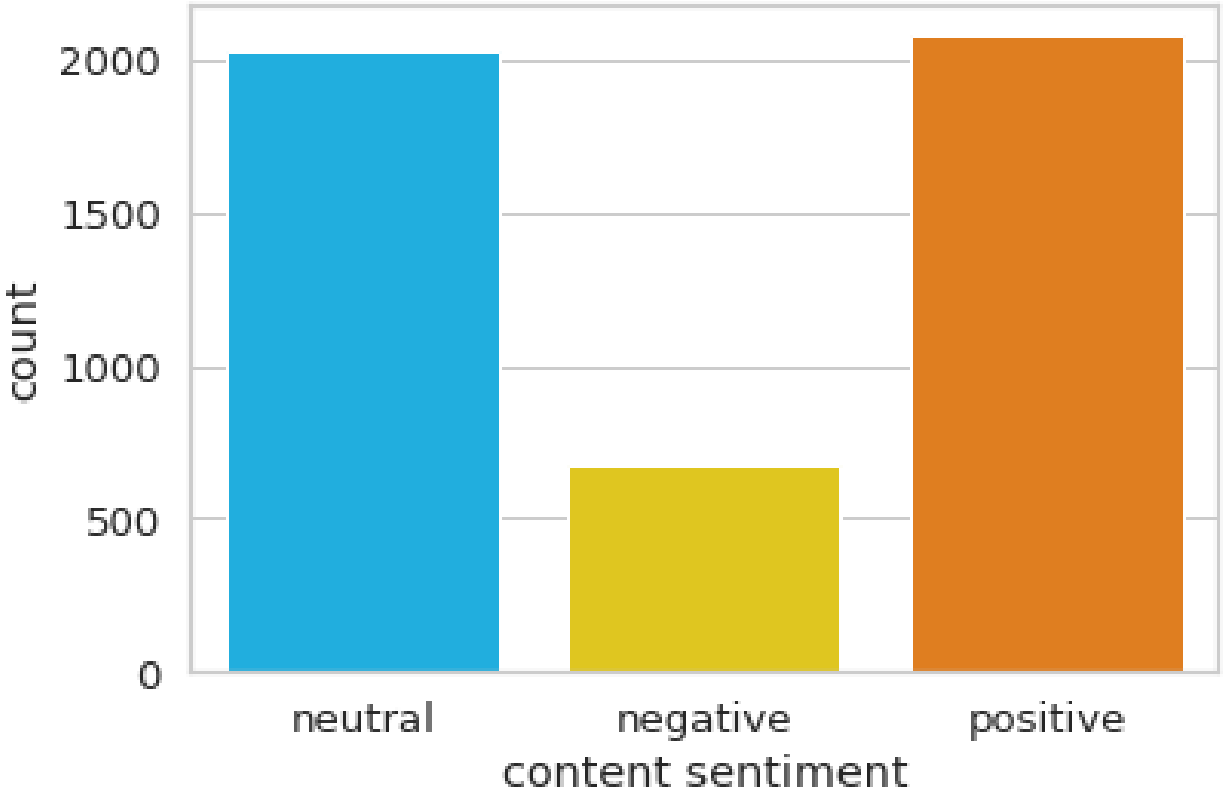
	precision	recall	f1-score	support
neutral	0.98	0.98	0.98	766
negative	0.60	0.60	0.60	5

positive	0.92	0.92	0.92	186
accuracy			0.97	957
macro avg	0.83	0.83	0.83	957
weighted avg	0.97	0.97	0.97	957

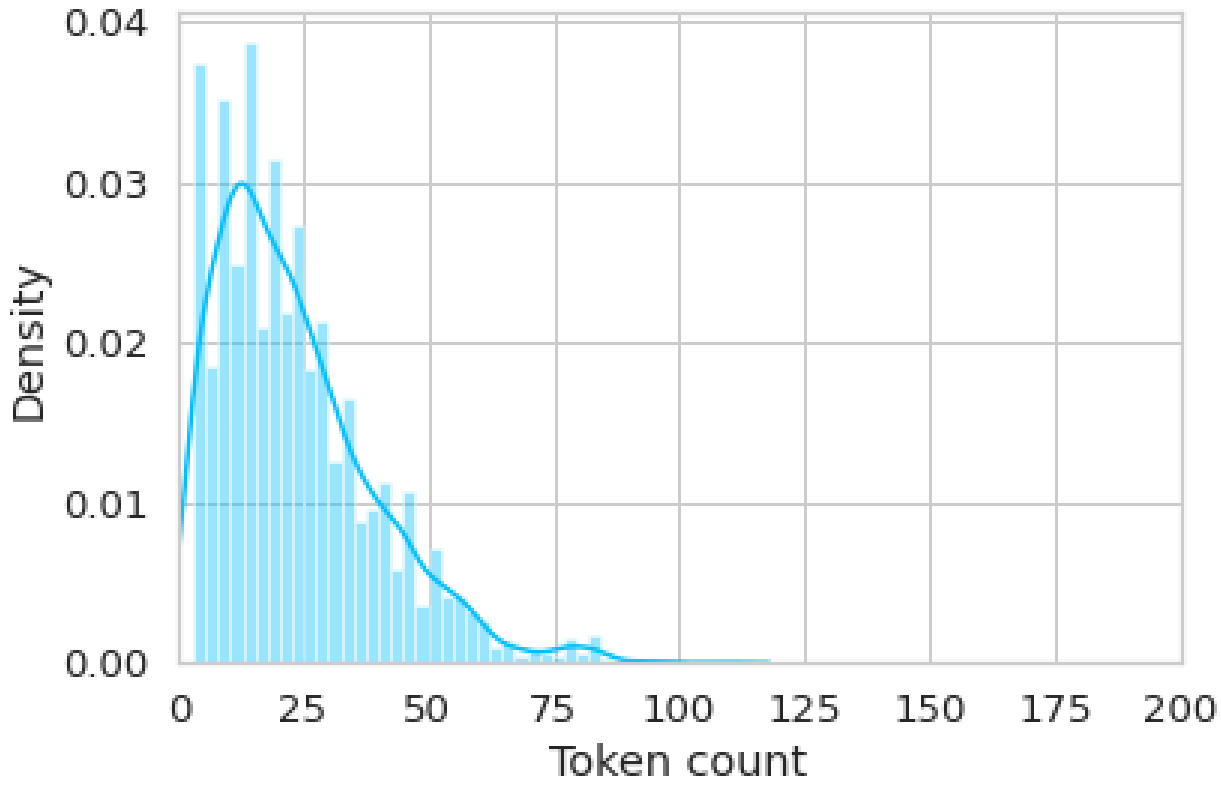
3.4. Analisi di NLP senza text-preprocessing

E' stato effettuato un processamento dei dati all'interno del quale sono state evitate le trasformazioni testuali, usate nei due modelli precedenti per ridurre la complessità del testo. Per quanto riguarda il sentiment calcolato sui dati testuali abbiamo riscontrato differenti risultati, i quali hanno prodotto differenti label in termini di occorrenze.

Ecco una rappresentazione grafica delle occorrenze del sentiment:



Anche ovviamente la lunghezza relativa alle singole frasi testuali risulta essere maggiore senza le trasformazioni effettuate sulle singole frasi. Ecco la rappresentazione grafica:



I risultati ottenuti sono i seguenti, a seguito della valutazione del classification report:

	precision	recall	f1-score	support
neutral	0.96	0.99	0.98	423
negative	0.95	0.89	0.92	144
positive	0.97	0.96	0.97	390
accuracy			0.96	957
macro avg	0.96	0.95	0.95	957
weighted avg	0.96	0.96	0.96	957

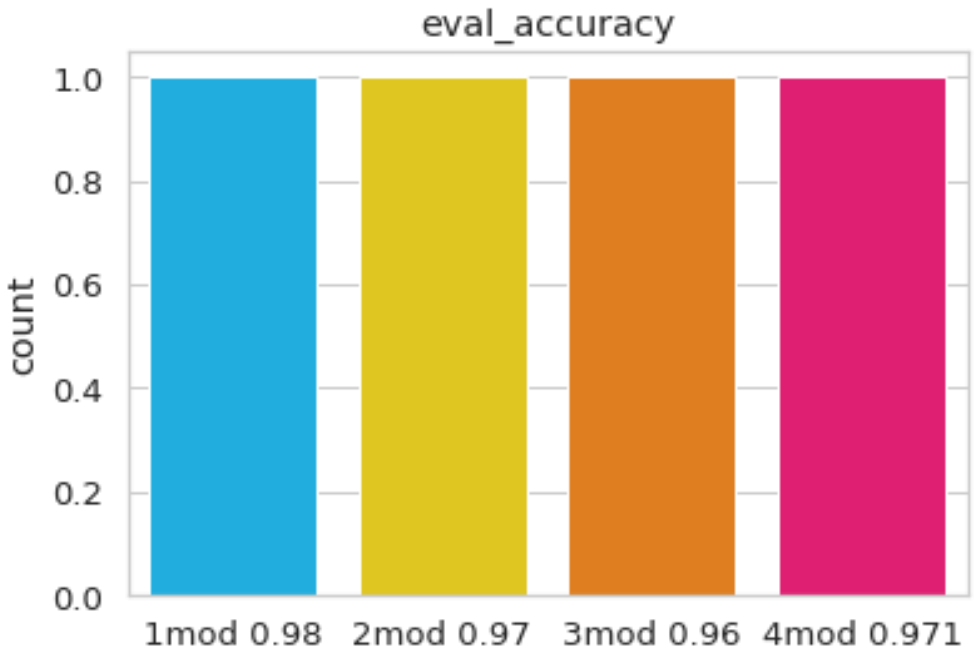
I risultati ottenuti, invece, a seguito della valutazione del classification report utilizzando come label il sentiment delle emoji è il seguente:

	precision	recall	f1-score	support
neutral	0.98	0.99	0.98	766
negative	0.75	0.60	0.67	5
positive	0.95	0.91	0.93	186
accuracy			0.97	957
macro avg	0.89	0.83	0.86	957
weighted avg	0.97	0.97	0.97	957

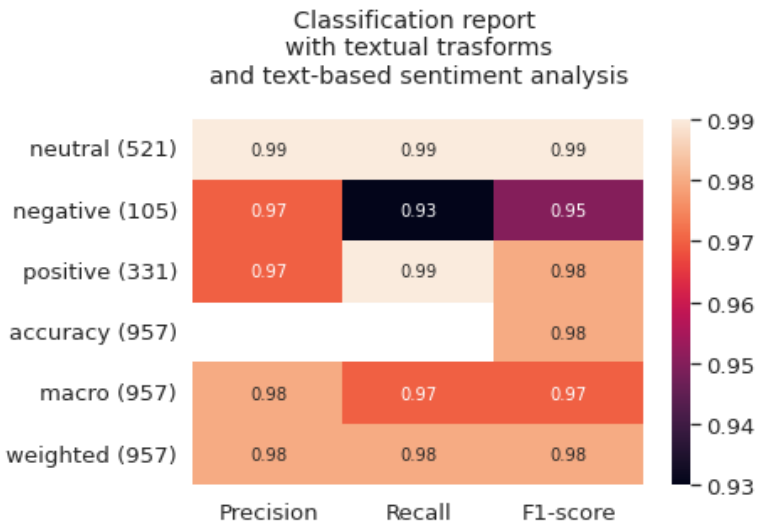
4. Valutazione

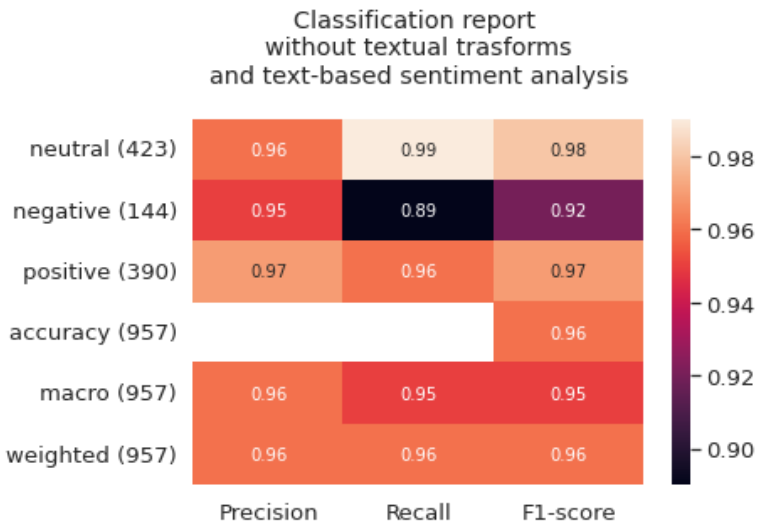
Una valutazione grafica sul valore di accuracy, riguardanti i 4 modelli, esposti precedentemente, in ordine, utilizzando il modello pre-addestrato RoBERTa:

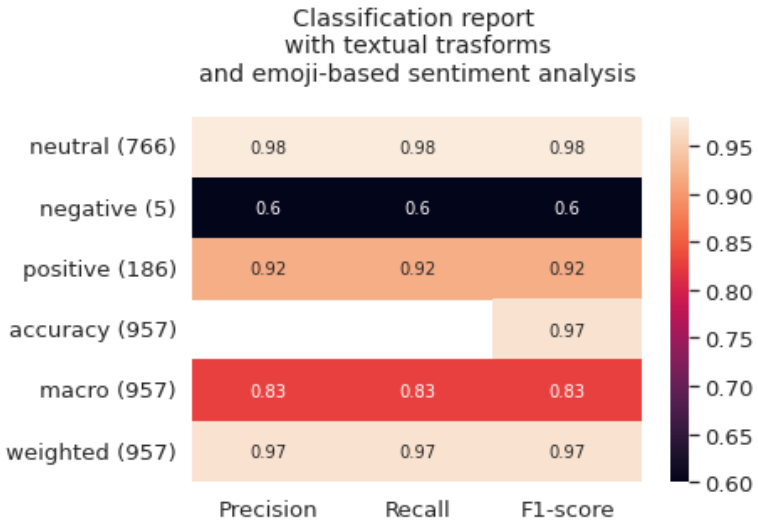
1. In blu, abbiamo il modello con preprocessing testuale e utilizzo come label il sentiment del testo.
2. In giallo, abbiamo il modello con preprocessing testuale e utilizzo come label il sentiment delle emoji.
3. In arancio, abbiamo il modello senza preprocessing testuale e utilizzo come label il sentiment del testo
4. In viola, abbiamo il modello senza preprocessing testuale e utilizzo come label il sentiment delle emoji.

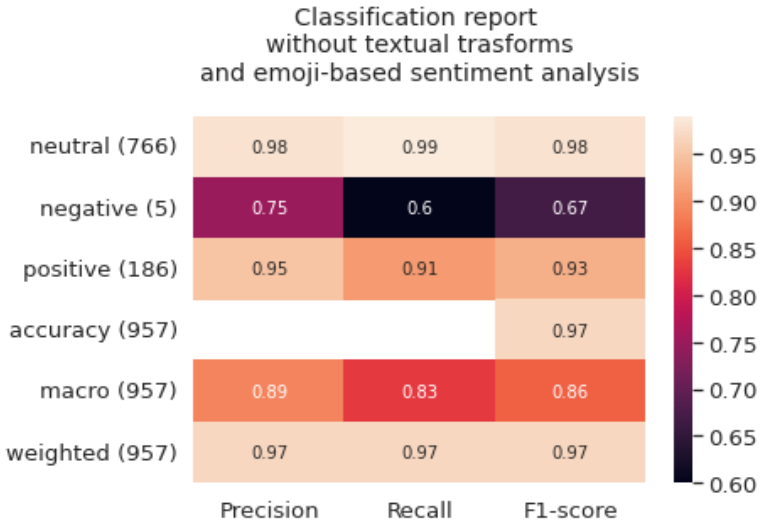


Valutazione del classification_report sfruttando il modello RoBERTa:









Infine, ho ripetuto i 4 modelli sfruttando questa volta l'algoritmo di Bert, nello specifico utilizzando un TokenizerFast con "bert-base-uncased", e ho effettuato una valutazione sui classification report per visualizzare dei risultati da poter mettere a confronto con i precedenti modelli addestrati attraverso RoBERTa.

Ecco le valutazioni del classification_report riscontrate: