A MULTISTAGE PIPELINE FOR DETECTING OFFENSIVE CONTENT IN SPOKEN LANGUAGE

Università degli Studi di Salerno
Dipartimento di Informatica
a.a. 2024-2025

Prof. Polese Giuseppe
Dott. Solimando Giandomenico
Grauso Teodoro
Maselli Mariantonietta
`https://github.com/grauso-t/Offensive-Language-Detector-Threads`

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

Negli ultimi anni, le piattaforme di social media come **Tiktok** e **Threads** hanno registrato una crescita esponenziale in termini di utenti e contenuti condivisi quotidianamente. Questi contenuti multimediali, spesso costituiti da video brevi con audio associato, rappresentano una fonte ricca di informazioni, ma possono anche veicolare messaggi potenzialmente dannosi o offensivi.

Il presente progetto si propone di sviluppare un sistema automatizzato in grado di analizzare contenuti testuali e audio estratti da video pubblicati su tali piattaforme, mediante tecniche di **speech-to-text**, partendo da un link fornito dall'utente. L'obiettivo principale è trasformare il flusso audio in testo scritto, per realizzarlo verranno applicati diversi algoritmi di machine learning e metodi di analisi del linguaggio naturale (NLP), a partire da modelli classici come la **regressione logistica** e le **Support Vector Machines** (SVM), fino a tecniche più avanzate di deep learning, quali il fine-tuning di modelli pre-addestrati come BERT [1]. Infine, si esplorerà l'uso di **modelli linguistici di grandi dimensioni** (LLM) per una comprensione semantica più profonda del testo.

La capacità di riconoscere e classificare i contenuti dannosi non solo contribuisce a migliorare l'esperienza degli utenti, ma rappresenta un passo fondamentale nella lotta contro la diffusione di discorsi di odio, discriminazione e altre forme di abuso online. Il sistema, quindi, si focalizza non solo sull'identificazione della natura dannosa del contenuto, ma anche sulla categorizzazione in specifiche tipologie di offesa quali **omofobia**, **sessismo** o **razzismo**.

2 Research Questions

In questa sezione definiamo le principali domande di ricerca che guideranno lo sviluppo e la valutazione del sistema automatizzato per l'analisi di contenuti audio estratti da piattaforme come **TikTok** [2] e **Threads** [3]. Tali RQ risultano fondamentali per identificare le sfide tecniche e valutare l'efficacia delle metodologie adottate.

Research Question 2.1. Quali modelli di *machine learning* offrono le migliori performance nella classificazione di contenuti offensivi e discriminatori nel testo trascritto?

L'obiettivo è confrontare modelli classici e avanzati, come la regressione logistica, le SVM e modelli basati su BERT, per identificare quello più efficace nel riconoscimento e nella categorizzazione delle diverse tipologie di contenuti dannosi.

Research Question 2.2. In che modo i *modelli linguistici di grandi dimensioni* (LLM) possono migliorare la comprensione semantica e la classificazione dei contenuti offensivi rispetto ai modelli tradizionali?

Questa domanda esplora il potenziale dei LLM nel cogliere sfumature linguistiche più complesse e nel ridurre falsi positivi/negativi nell'identificazione dei contenuti.

3 Metodologia

3.1 Dataset

Per il nostro studio, sono stati utilizzati quattro dataset principali, ognuno dei quali contiene esempi di contenuti testuali classificati in base alla presenza di contenuti offensivi o discriminatori. I dataset provengono da diverse fonti e trattano argomenti quali omofobia, sessismo, razzismo e contenuti neutrali. Di seguito vengono descritti i passaggi di preprocessing e la preparazione dei dati per il successivo utilizzo nei modelli di machine learning.

Raccolta dei Dati I dati provengono da due principali fonti pubbliche e sono stati utilizzati per ciascuna delle categorie di offesa analizzate:

- **Omofobia**: Il dataset relativo a contenuti omofobi è stato scaricato da **Hugging Face**[4] e contiene un insieme di tweet etichettati come contenenti o meno omofobia. La versione utilizzata è disponibile al repository `JoshMcGiff/HomophobiaDetectionTwitterX` [5].
- **Sessismo**: Il dataset relativo a contenuti sessisti è stato scaricato da **Kaggle**[6] e include testi etichettati come sessisti o non sessisti. La versione utilizzata è *Sexism Detection in English Texts* di *Aadyasingh55* [7].
- **Razzismo**: Il dataset relativo a contenuti razzisti proviene da Kaggle e contiene tweet annotati come razzisti o non razzisti. La versione utilizzata è *Racism Tweets* di *Raghad Abdullah* [8].
- **Contenuti Neutrali**: Il dataset dei contenuti neutrali è stato generato artificialmente tramite un **modello di linguaggio di grandi dimensioni** (LLM), in particolare **ChatGPT** di *OpenAI* [9], al fine di creare esempi di testo non offensivi e bilanciare le classi presenti nel dataset.

Preprocessing dei Dati I dati sono stati sottoposti a una serie di operazioni di pulizia per rimuovere caratteri non rilevanti e preparare i testi per la successiva fase di tokenizzazione e vettorizzazione. Le seguenti operazioni sono state applicate a ciascun dataset:

- Rimozione di URL e menzioni di utenti (*@username*).
- Rimozione di hashtag.
- Rimozione di entità HTML e caratteri non ASCII.
- Normalizzazione del testo, inclusa la rimozione di ripetizioni di lettere (es. "cooool" $\rightarrow$ "cool").
- Conversione del testo in minuscolo.

Dopo queste operazioni di pulizia, il testo è stato pronto per essere utilizzato come input nei modelli di machine learning.

Bilanciamento e Unione dei Dataset Per affrontare eventuali squilibri nel numero di campioni tra le categorie, è stato eseguito un *undersampling* delle classi maggioritarie, riducendo il numero di esempi delle classi più rappresentate fino a ottenere un dataset bilanciato. Successivamente, i dataset di omofobia, sessismo, razzismo e contenuti neutrali sono stati concatenati in un unico dataset, che è stato ulteriormente bilanciato.

Il dataset finale è stato quindi suddiviso in due set: uno di **addestramento** e uno di **test**, utilizzando una proporzione dell'80% per l'addestramento e del 20% per il test.

Il dataset finale bilanciato è stato poi salvato in formato `.csv` per un facile utilizzo durante le fasi di addestramento e valutazione dei modelli.

Statistiche del Dataset Il dataset bilanciato contiene **621** campioni per ciascuna classe come visualizzato nella Figura 1. Tutti i campioni sono stati etichettati correttamente e sono pronti per essere utilizzati nei modelli di classificazione.

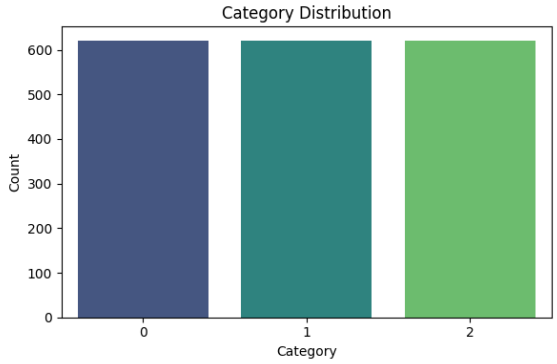


Figura 1: Distribuzione delle categorie

3.2 Classificazione

Il processo di classificazione del contenuto testuale si articola in due fasi distinte, seguendo un approccio gerarchico e modulare volto a massimizzare l'efficienza e la precisione.

Fase 1: Rilevamento del contenuto offensivo In un primo momento, ogni testo in input viene tradotto in lingua inglese e sottoposto a una pulizia e pre-processing. Successivamente, il contenuto viene valutato attraverso un classificatore preaddestrato di tipo *toxic-BERT* [10], specializzato nell'individuazione di contenuti tossici. Il modello restituisce una serie di etichette binarie associate a punteggi di confidenza per varie categorie di tossicità, tra cui: *toxicity*, *severe toxicity*, *obscene*, *identity attack*, *insult*, *threat*, *sexual explicit*.

Se almeno una di queste etichette supera una soglia predefinita, il testo viene considerato potenzialmente offensivo e viene inoltrato alla seconda fase per una classificazione più specifica. In caso contrario il contenuto viene ignorato, evitando inutili computazioni.

Fase 2: Classificazione specifica della tipologia di offesa Una volta identificato come offensivo, il testo viene eventualmente vettorializzato tramite **TF-IDF** e classificato in una delle seguenti categorie:

- **Omofobia** (*homophobia*)
- **Sessismo** (*sexism*)
- **Razzismo** (*racism*)
- **Altro contenuto offensivo** (*offensive content*)

La classificazione avviene tramite più modelli supervisionati addestrati sul dataset bilanciato. I modelli impiegati includono: **regressione logistica**, **SVM lineare**, **BERT fine-tuned**, e **modelli LLM generativi** per valutazione qualitativa.

Questo approccio consente di separare la componente generica di rilevamento da quella specifica di classificazione, migliorando la scalabilità del sistema e facilitando la manutenzione modulare. Inoltre, riduce il numero di falsi positivi e permette di utilizzare modelli più complessi solo quando necessario.

Pseudocodice del processo

```
Input: testo
traduci -> pulisci -> verifica tossicità (toxic-BERT)
  |
  |--- se non tossico -> esci
  |
  |--- se tossico:
        -> scegli modello:
            |
            |--- Logistic/SVM:
            |       -> vettorializza (TF-IDF)
            |       -> classifica
            |
            |--- BERT/LLM:
                    -> classifica
        -> restituisci categoria specifica
```

Questa strategia garantisce un buon compromesso tra precisione, efficienza e possibilità di aggiornamento incrementale dei modelli classificatori.

3.3 Logistic Regression

Per l'addestramento di un classificatore lineare, si è fatto uso della **regressione logistica** [11], una tecnica largamente impiegata per problemi di classificazione binaria e multiclasse. La regressione logistica è un modello di classificazione lineare che stima la probabilità che un'osservazione appartenga a una determinata classe. Applica una funzione sigmoide alla combinazione lineare delle feature in input, producendo un output compreso tra 0 e 1. L'addestramento avviene minimizzando la *log-loss*, penalizzando le previsioni errate in base alla loro incertezza.

Preprocessing, Bilanciamento del Dataset e Rappresentazione dei Dati Il dataset utilizzato è stato sottoposto a una fase di pulizia e bilanciamento, includendo unicamente le istanze appartenenti alle tre categorie di offesa: *omofobia*, *sessismo* e *razzismo*. Le istanze appartenenti alla categoria generica "altro" sono state escluse per focalizzare l'analisi su classi ben definite. Dopo questa operazione, la distribuzione delle classi è stata controllata e visualizzata per garantire un dataset bilanciato e adatto all'addestramento supervisionato.

I testi trascritti sono stati suddivisi in *training set* (80%) e *test set* (20%), mantenendo la proporzione originale delle classi tramite stratificazione. Per trasformare i testi in una rappresentazione numerica adatta ai modelli di apprendimento automatico, è stato utilizzato il metodo **TF-IDF (Term Frequency - Inverse Document Frequency)** [12], con un massimo di 1000 feature, includendo uni-grammi e bi-grammi e rimuovendo parole comuni in lingua inglese (*stop words*).

Modello di Regressione Logistica Il modello utilizzato per questa fase iniziale di classificazione è la **regressione logistica multinomiale**, un metodo lineare interpretabile che permette di stimare la probabilità che un dato testo appartenga a ciascuna classe. L'algoritmo è stato addestrato utilizzando i dati vettorializzati e ottimizzato tramite la discesa del gradiente (*Gradient Descent*), con un numero massimo di iterazioni pari a 1000 per garantire la convergenza.

Valutazione del Modello Le performance del modello sono state valutate sull'insieme di test tramite diverse metriche: ***accuracy***, ***precision***, ***recall*** e ***F1-score*** per ciascuna classe. Inoltre, è stata prodotta una **matrice di confusione** per visualizzare le predizioni corrette e gli errori di classificazione per ciascuna categoria.

Analisi delle feature Per approfondire l'analisi e migliorare l'interpretabilità del modello, sono stati visualizzati i vocaboli con i pesi più alti nel vettore dei coefficienti della regressione logistica per ciascuna classe. Questi pesi rappresentano l'importanza relativa dei termini nella classificazione di un testo come offensivo in una specifica categoria. Tali visualizzazioni offrono un'utile panoramica sui segnali linguistici più informativi per il modello.

Il modello addestrato, insieme al vettorizzatore TF-IDF, è stato salvato per consentire la futura applicazione su nuovi dati e garantire la riproducibilità dell'esperimento. I risultati della valutazione sono stati inoltre esportati in formato `JSON`, rendendo possibile l'integrazione con altri moduli del sistema.

3.4 Linear SVM

Successivamente, è stato adottato un approccio supervisionato basato su **Support Vector Machine lineare** (Linear SVM), implementato tramite la libreria `scikit-learn` [13]. È una tecnica di apprendimento supervisionato utilizzata per compiti di classificazione e regressione. Nella variante usata in questo lavoro, la **Support Vector Classification** (SVC), il modello cerca di trovare il margine massimo che separa le classi nel dataset. Solo un sottoinsieme di esempi (**vettori di supporto**) contribuisce a determinare il confine decisionale, rendendo il modello particolarmente efficace anche in presenza di dati ad alta dimensionalità. Il dataset utilizzato è stato caricato da file `.csv` e filtrato per rimuovere le istanze con categoria etichettata come `3` (contenuto neutrale).

Preprocessing e Rappresentazione dei Dati Anche in questo caso i testi sono stati rappresentati utilizzando la **vettorizzazione TF-IDF (Term Frequency - Inverse Document**

Frequency), limitando il numero massimo di feature a 1000 e considerando n-grammi di ordine 1 e 2 (unigrammi e bigrammi). Sono state rimosse le *stop wording* in lingua inglese. Il dataset è stato suddiviso in *training set* (80%) e *test set* (20%), mantenendo la proporzione delle classi mediante *stratified sampling*.

Modello di classificazione Il modello adottato è una **Linear SVM** con parametro `max_iter=1000` e `random_state=42` per garantire ripetibilità. Il classificatore è stato addestrato sui vettori TF-IDF e ha appreso un iperpiano separatore per ogni classe, massimizzando il margine tra le classi.

Analisi delle feature Per ciascuna classe, sono stati identificati i **10 n-grammi con peso maggiore** nei vettori di coefficiente della SVM. Questi termini rappresentano quelli più fortemente associati alla rispettiva categoria di hate speech. I pesi sono stati visualizzati mediante grafici a barre. Il modello addestrato e il vettorizzatore sono stati serializzati tramite `joblib` e salvati in formato `.pkl`. I risultati della valutazione sono stati salvati in formato `JSON` per un facile riutilizzo.

3.5 Fine-Tuning di BERT

In questa sezione descriviamo il processo di fine-tuning del modello BERT per la classificazione di testi in quattro categorie, utilizzando PyTorch e la libreria `transformers` [14]. Il fine-tuning di BERT consiste nell'adattare un modello pre-addestrato su grandi raccolte di testi a un compito specifico, come la classificazione del testo. In questa fase, si aggiornano i pesi del modello tramite ulteriore addestramento supervisionato su un dataset etichettato, mantenendo però la conoscenza linguistica acquisita durante il pretraining. Questo approccio consente di ottenere alte prestazioni anche con quantità di dati relativamente contenute.

Caricamento e preparazione dei dati I dati vengono caricati e filtrati per le categorie di interesse, e suddivisi in *training* e *validation set* tramite `train_test_split`. Successivamente si applica la tokenizzazione con il tokenizer di BERT `bert-base-uncased` e si crea un dataset custom PyTorch per gestire gli input.

Addestramento del modello Si carica il modello `BertForSequenceClassification` pre-addestrato, specificando il numero di classi (4) per la classificazione.

Gli argomenti per il training sono definiti tramite `TrainingArguments`, con:

- 5 epoche di addestramento,
- `batch size` pari a 8 per training e validation,
- `warmup steps`, `weight decay` e `fp16` per ottimizzare le prestazioni,
- disabilitato il salvataggio automatico con `save_strategy="no"` per permettere un controllo manuale.

L'addestramento avviene tramite un ciclo manuale di epoche, che chiama `trainer.train()` per ogni epoca. Durante ogni epoca si raccolgono le loss di training e validation, valutando il modello manualmente con `trainer.evaluate()`. Vengono stampate le metriche principali dopo ogni epoca.

Valutazione finale e salvataggio risultati Al termine del training si esegue la valutazione finale sul validation set usando `trainer.predict()`, da cui si estraggono predizioni e label. Si calcolano metriche aggregate, il report di classificazione e la matrice di confusione. Il modello e il tokenizer vengono salvati manualmente nelle rispettive cartelle per poter essere riutilizzati successivamente. Infine viene generato e salvato un grafico con l'andamento delle loss di training e validation durante le epoche, utile per analizzare la convergenza del modello.

3.6 Utilizzo di ChatGPT per la classificazione del contenuto offensivo

Per sfruttare le capacità contestuali avanzate dei **modelli linguistici di grandi dimensioni** (Large Language Models, LLM), abbiamo integrato `gpt-3.5-turbo` di OpenAI [9] nel nostro pipeline di classificazione. L'obiettivo era verificare se un LLM potesse migliorare la rilevazione del contenuto offensivo cogliendo sfumature linguistiche che spesso sfuggono ai modelli statistici o ai *Transformer* addestrati con supervisione.

Architettura del prompt e logica di inferenza Il funzionamento si basa sull'interazione con il modello via API attraverso l'endpoint `chat/completions`. Il prompt inviato al sistema è costituito da due messaggi:

- Un messaggio di sistema che imposta il ruolo del modello: *"You are an offensive sentence classifier. Respond only with one of these words: 'no offensive content', 'sexism', 'racism', 'homophobia'."*
- Un messaggio utente contenente la frase da classificare nel formato: *"Sentence: <testo >"*

In questo modo, si forza il modello a fornire una risposta chiusa, scegliendo esclusivamente tra le quattro etichette target.

Classificazione diretta A differenza degli altri modelli usati (come BERT fine-tuned o regressori tradizionali), il modello GPT non è stato riaddestrato o adattato: abbiamo sfruttato la sua capacità di **few-shot learning** in modalità **zero-shot**. Il modello produce una classificazione pur non essendo stato specificamente addestrato sul nostro dataset, basandosi unicamente sulle istruzioni fornite nel prompt.

3.7 Utilizzo di Mistral per la classificazione del contenuto offensivo

Oltre all'integrazione con modelli ospitati via API (come `gpt-3.5-turbo`), è stato sperimentato l'utilizzo di un modello linguistico locale basato su architettura LLaMA, nello specifico una variante del modello `Mistral-7B` [15] distribuita tramite GGUF [16]. Il caricamento e l'inferenza sono stati effettuati localmente sfruttando librerie compatibili come `llama-cpp-python` e wrapper ad alte prestazioni come `ollama` [17].

Motivazione Questa scelta è motivata dalla necessità di garantire maggiore privacy e controllo sui dati, evitando il trasferimento di contenuti sensibili su server remoti. Inoltre, la possibilità di eseguire inferenze in locale consente l'elaborazione su larga scala e riduce la dipendenza da infrastrutture esterne, rendendo il sistema più flessibile e replicabile.

Prompt engineering e logica di classificazione Analogamente all'approccio adottato con GPT, il modello locale riceve in input un prompt strutturato che imposta chiaramente il compito di classificazione. In particolare, il testo (dopo essere stato tradotto in inglese e pulito) viene inserito in un prompt del tipo: *You are an offensive sentence classifier. Classify the following sentence into one of the following categories: 'no offensive content', 'sexism', 'racism', or 'homophobia'. Respond with just one of these categories.*

L'output del modello consiste in una singola etichetta tra quelle previste, del tipo *Sentence: <testo >Category:.* Per garantire la robustezza della classificazione, viene disabilitata la temperatura stocastica (`temperature = 0`), così da ottenere risposte deterministiche.

Integrazione nel sistema Il modello locale è caricato dinamicamente a seconda del tipo di dispositivo disponibile (CPU o GPU), con un numero di layer ottimizzato in base alla memoria. Non è richiesto alcun addestramento aggiuntivo: il modello è utilizzato in modalità *zero-shot*, sfruttando la conoscenza già appresa durante il pretraining.

Vantaggi e limiti Il principale vantaggio risiede nella possibilità di controllare completamente l'ambiente di esecuzione, riducendo costi e dipendenze. Tuttavia, i LLM locali possono essere meno aggiornati o meno performanti rispetto alle controparti commerciali su cloud, e richiedono un'ottimizzazione attenta delle risorse hardware disponibili.

3.8 Acquisizione e Preprocessing dei Dati Multimediali

Per supportare l'utente nell'analisi di nuovi contenuti multimediali provenienti da piattaforme social, è stata implementata una procedura di estrazione e trascrizione di dati audio e testuali. È importante sottolineare che i dati acquisiti tramite questa procedura non sono utilizzati per l'addestramento dei modelli, che si basano esclusivamente sui dataset precedentemente raccolti e pre-elaborati.

Download e Estrazione Audio da TikTok È stato sviluppato uno script basato sulla libreria `yt_dlp` per scaricare l'audio dai video di **TikTok**[2]. Il sistema:

- Verifica la validità dell'URL come link TikTok.
- Effettua tentativi multipli di download utilizzando cookie di diversi browser per superare eventuali limitazioni.
- Estrae e salva temporaneamente l'audio in formato MP3 per la successiva trascrizione.
- Elimina il file temporaneo dopo aver acquisito il contenuto audio.

Estrazione del Testo da Threads tramite Web Scraping Per i post di **Threads**, si è utilizzato un approccio di web scraping con `Selenium` [18] e `BeautifulSoup` [19] che:

- Carica la pagina in modalità *headless* per automatizzare la navigazione.
- Identifica e analizza gli elementi HTML contenenti autore, data e testo del post.
- Gestisce eventuali errori dovuti a modifiche nella struttura della pagina o assenza di dati.

Trascrizione Audio con Whisper I file audio estratti vengono trascritti in testo tramite il modello `Whisper` di OpenAI [20], un sistema di riconoscimento vocale automatico avanzato, per consentire un'analisi testuale immediata da parte dell'utente.

Questa pipeline serve esclusivamente come strumento di supporto per l'utente nella fase di controllo e analisi manuale di nuovi contenuti, mentre l'addestramento dei modelli predittivi avviene unicamente sui dataset originali elaborati nella fase precedente.

Considerazioni e Criticità Tecniche Durante la progettazione di questa pipeline sono emerse alcune difficoltà operative, legate soprattutto alla natura delle piattaforme da cui si estraggono i contenuti. L'utilizzo di cookie di browser diversi si è reso necessario per superare restrizioni dinamiche applicate da TikTok, ma può risultare instabile e richiedere aggiornamenti frequenti. Allo stesso modo, l'approccio tramite web scraping con Selenium e BeautifulSoup, sebbene efficace, è soggetto a rottura nel momento in cui la struttura HTML dei siti cambia, rendendo necessaria una manutenzione continua dello script.

Infine, anche la gestione dei file temporanei e delle trascrizioni comporta attenzione nella pulizia e ottimizzazione delle risorse locali, specialmente in scenari di uso ripetuto. Nonostante queste criticità, la pipeline si è dimostrata funzionale e adattabile per il supporto all'analisi manuale di contenuti, mantenendo la flessibilità necessaria a un contesto sperimentale.

4 Risultati

4.1 RQ1

Research Question 4.1. Quali modelli di *machine learning* offrono le migliori performance nella classificazione di contenuti offensivi e discriminatori nel testo trascritto?

4.1.1 Logistic Regression

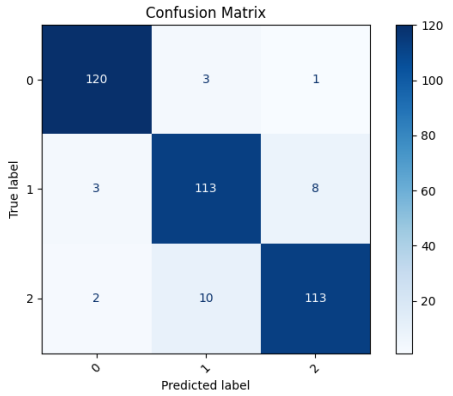
Abbiamo valutato il modello di regressione logistica sul nostro dataset bilanciato, ottenendo un'accuratezza complessiva (*accuracy*) del **92.76%**. Di seguito riportiamo una tabella riassuntiva con le principali metriche di performance suddivise per classe.

Tabella 1: Metriche di classificazione per il modello di Regressione Logistica

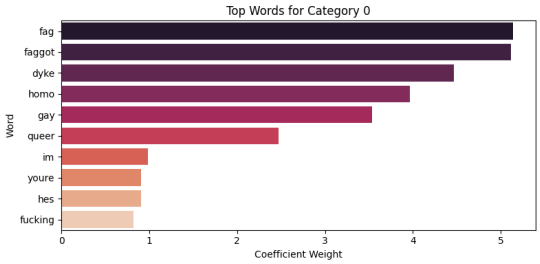
Classe	Precision	Recall	F1-score	Support
0 (Omofobia)	0.96	0.968	0.964	124
1 (Sessismo)	0.897	0.911	0.904	124
2 (Razzismo)	0.926	0.904	0.915	125
Macro Avg	0.928	0.928	0.928	373
Weighted Avg	0.928	0.928	0.928	373

La matrice di confusione (Figura 2a) evidenzia le prestazioni del modello nel distinguere correttamente le diverse classi, mostrando un buon bilanciamento tra precision e recall.

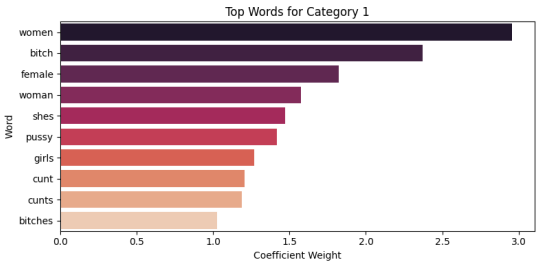
In aggiunta, le figure associate mostrano le parole più significative per ogni categoria, basate sui coefficienti del modello. Queste parole rappresentano i segnali testuali più forti che la regressione logistica ha utilizzato per classificare i testi nelle rispettive categorie.



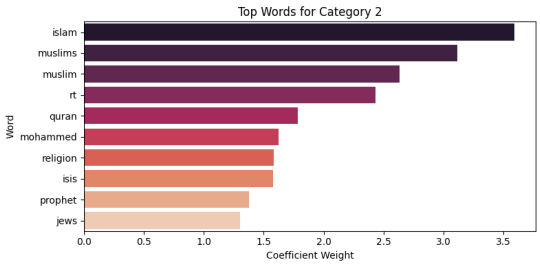
(a) Matrice di confusione



(b) Top parole - Categoria 0 (Omofobia)



(c) Top parole - Categoria 1 (Sessismo)



(d) Top parole - Categoria 2 (Razzismo)

Figura 2: Analisi visiva dei risultati del modello Regressione Logistica

Sintesi e conclusioni sui risultati I risultati mostrano come la regressione logistica, pur essendo un modello relativamente semplice, riesca a catturare pattern significativi nei dati testuali per discriminare efficacemente tra le categorie di interesse. Il modello di regressione logistica ha mostrato ottime prestazioni su dataset bilanciato, con un'accuracy complessiva del **92.76%**. La matrice di confusione indica che le tre classi principali sono state distinte correttamente nella maggior parte dei casi, senza particolari bias tra precision e recall. Le parole salienti (*top features*) per ciascuna classe risultano semanticamente coerenti e interpretabili, suggerendo che il modello basa le sue decisioni su segnali linguistici significativi.

Nonostante queste performance promettenti, occorre tener presente che la regressione logistica ha una capacità limitata nel comprendere contesti complessi, sfumature idiomatiche o ironiche. Questo rende necessario affiancarla a modelli più sofisticati (come BERT o LLM) per compiti di classificazione che richiedono un'analisi profonda del linguaggio.

Tuttavia, il suo utilizzo ha offerto un'ottima baseline interpretativa, veloce da addestrare e da spiegare, ideale per sistemi ibridi in cui un primo filtro leggero viene seguito da un modello più potente per i casi ambigui.

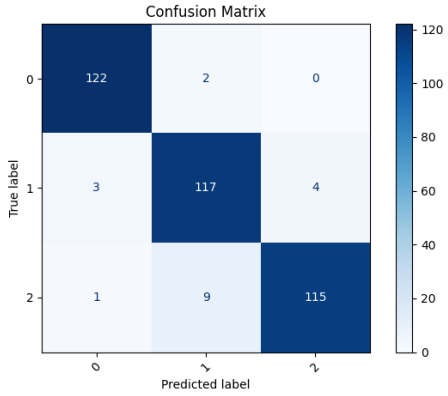
4.1.2 Linear SVM

Il modello SVM lineare ha ottenuto risultati superiori rispetto alla regressione logistica, raggiungendo un'accuracy del **94.91%**. Come si evince dalla Tabella 2, tutte e tre le classi vengono identificate con precision e recall molto elevate, in particolare la classe 0 (Omofobia), per cui il modello mostra una quasi perfetta capacità di classificazione.

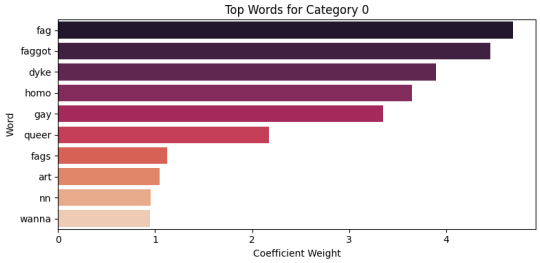
Tabella 2: Metriche di classificazione per il modello SVM Lineare

Classe	Precision	Recall	F1-score	Support
0 (Omofobia)	0.968	0.984	0.976	124
1 (Sessismo)	0.914	0.944	0.929	124
2 (Razzismo)	0.966	0.920	0.943	125
Macro Avg	0.950	0.949	0.949	373
Weighted Avg	0.950	0.949	0.949	373

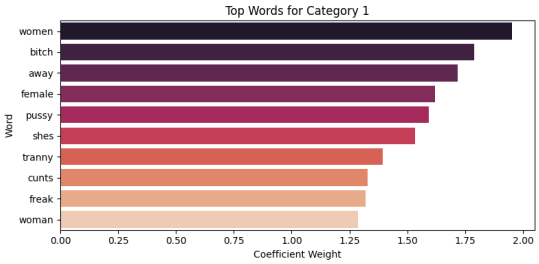
La matrice di confusione conferma un'elevata capacità del modello di distinguere correttamente tra le classi. In particolare la classe **0** viene riconosciuta quasi perfettamente, con un f1-score di **0.976**. Le parole con i coefficienti più alti per ciascuna classe risultano coerenti e rappresentative dei contenuti specifici delle categorie, indicando una buona interpretabilità del modello. La combinazione di alta performance e trasparenza lo rende un candidato robusto per la classificazione automatica di contenuti.



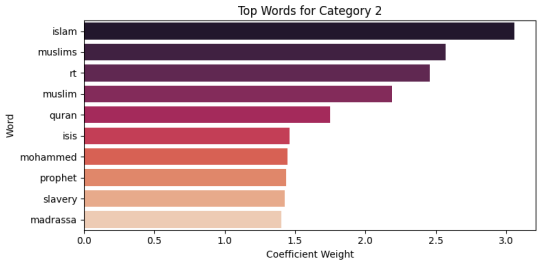
(a) Matrice di confusione



(b) Top parole per la classe 0 (Omofobia)



(c) Top parole per la classe 1 (Sessismo)



(d) Top parole per la classe 2 (Razzismo)

Figura 3: Analisi visiva dei risultati del modello SVM Lineare

Sintesi e conclusioni sui risultati Il modello SVM lineare ha mostrato prestazioni migliori rispetto alla Regressione Logistica, con un'accuracy complessiva pari al **94.91%**, migliorando di circa due punti percentuali. In particolare, si osserva un incremento significativo dell'*F1-score* per la classe 1 (**0.929** rispetto a **0.904**), indicando una gestione più efficace delle ambiguità lessicali associate a questa categoria.

Anche la precision e il recall medi (macro e pesati) risultano elevati e molto bilanciati, suggerendo che il modello non soffra di squilibri tra le classi. Inoltre, l'analisi delle top features evidenzia una selezione coerente e discriminativa dei termini più informativi per ciascuna categoria, rafforzando la robustezza dell'approccio SVM.

Nel complesso, il modello SVM appare più adatto a questo compito di classificazione testuale, grazie alla sua capacità di catturare confini decisionali netti e ben separabili anche in spazi ad alta dimensionalità, come quelli derivati da rappresentazioni TF-IDF.

4.1.3 Fine-tuning di BERT

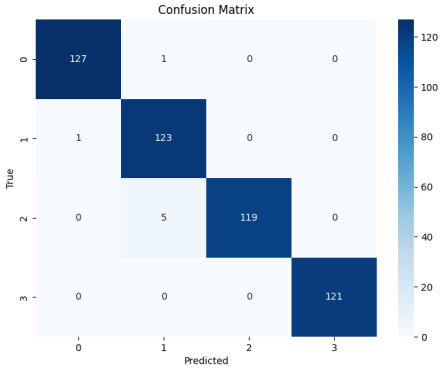
A differenza dei modelli classici precedentemente analizzati che sono stati addestrati unicamente per distinguere tra tipologie di contenuti offensivi, il modello BERT è stato addestrato anche a

riconoscere la **classe neutra**. Questo significa che il suo obiettivo non era soltanto identificare e classificare i comportamenti offensivi, ma anche discernere i casi in cui il contenuto del testo non presenta alcuna forma di aggressività o ostilità.

Questa impostazione più ampia e generalizzata rende il compito del modello significativamente più complesso, poiché richiede una comprensione più sottile e profonda del linguaggio, inclusi tono, contesto e ambiguità semantiche. Tuttavia, come si vedrà nei risultati, il modello BERT ha mostrato un'elevata capacità discriminativa su tutte le classi, incluse quelle non offensive, confermando l'efficacia del fine-tuning effettuato su una base pre-addestrata altamente sofisticata.

Tabella 3: Metriche di classificazione per il modello BERT fine-tuned

Classe	Precision	Recall	F1-score	Support
0 (Omofobia)	0.992	0.992	0.992	128
1 (Sessismo)	0.954	0.992	0.972	124
2 (Razzismo)	1.000	0.960	0.979	124
3 (Neutrale)	1.000	1.000	1.000	121
Macro Avg	0.986	0.986	0.986	497
Weighted Avg	0.986	0.986	0.986	497



(a) Matrice di confusione del modello BERT

(b) Andamento della training loss durante il fine-tuning

Figura 4: Analisi visiva dei risultati del modello BERT fine-tuned

Andamento della Training Loss

- **Inizio (Epoca 1)**: La Training Loss parte da un valore relativamente alto (circa 0,8), indicando che il modello inizialmente fatica a riconoscere pattern significativi nei dati di training.
- **Decremento rapido (Epoche 2–3)**: La loss decresce rapidamente verso valori prossimi allo zero, suggerendo un apprendimento efficace del modello sui dati di addestramento.

- **Stazionarietà (Epoche 3–4)**: Dopo la terza epoca, la Training Loss si stabilizza vicino a zero, indicando che il modello ha appreso in modo sufficiente le strutture presenti nei dati di training.
- **Aumento improvviso (Epoche 5–6)**: Nelle ultime epoche si osserva un aumento netto della Training Loss. Ciò potrebbe derivare da:
  - *Overfitting*: il modello inizia a memorizzare i dati di training, perdendo capacità di generalizzazione.
  - *Instabilità dell'ottimizzazione*: possibile fluttuazione nell'ottimizzatore verso la fine del training.

Andamento della Validation Loss

- **Inizio (Epoca 1)**: La Validation Loss inizia da un valore leggermente inferiore rispetto alla Training Loss (circa 0,6), indicando una buona capacità iniziale di generalizzazione.
- **Crescita progressiva (Epoche 2–6)**: La curva mostra un costante aumento, raggiungendo valori superiori a 1,2 nell'ultima epoca. Questo comportamento è indicativo di perdita di capacità predittiva sui dati di validazione.
- **Overfitting evidente**: L'aumento progressivo della Validation Loss, contrapposto alla stabilità o al calo della Training Loss, è un segnale chiaro di overfitting.

Sintesi e conclusioni sui risultati Il grafico della loss evidenzia un possibile fenomeno di **overfitting**, manifestato da una crescente divergenza tra la *Training Loss* e la *Validation Loss* nelle ultime epoche. Questo potrebbe indicare che il modello stia perdendo capacità di generalizzazione, o che il processo di ottimizzazione sia diventato instabile verso la fine.

Tuttavia, nonostante l'andamento della loss, il modello BERT fine-tunato ha ottenuto prestazioni eccezionali sul set di test, con un'accuracy del **98.59%** e valori di *precision*, *recall* e *f1-score* elevatissimi su tutte le classi. In particolare, sulla classe 3 (contenuto neutro), il modello raggiunge performance perfette, segno di una robusta comprensione del linguaggio naturale e delle sue sfumature semantiche.

Questa apparente discrepanza tra loss e metriche suggerisce che il modello, pur mostrando segni di overfitting nella fase finale, mantiene comunque una capacità predittiva molto elevata. Ciò rafforza l'idea che i modelli linguistici pre-addestrati come BERT, se fine-tunati correttamente, siano estremamente efficaci anche su compiti di classificazione testuale sensibili come il riconoscimento di contenuti offensivi.

4.2 RQ2

Research Question 4.2. In che modo i *modelli linguistici di grandi dimensioni* (LLM) possono migliorare la comprensione semantica e la classificazione dei contenuti offensivi rispetto ai modelli tradizionali?

4.2.1 ChatGPT

Per testare le potenzialità dei modelli linguistici di grandi dimensioni (LLM) nella classificazione di contenuti offensivi, abbiamo impiegato `gpt-3.5-turbo`, uno dei modelli più avanzati e accessibili forniti da OpenAI. Il modello è stato interrogato in modalità *zero-shot*, senza alcun riaddestramento, sfruttando esclusivamente un prompt progettato per forzare una risposta chiusa tra le quattro categorie target: *homophobia*, *sexism*, *racism*, e *no offensive content*.

Metriche di classificazione Il modello ha ottenuto un'accuracy complessiva del **77.5%** su un campione di 80 frasi, mostrando performance sorprendentemente solide per un approccio zero-shot. I risultati più elevati si osservano nella classe *racism*, che raggiunge una precision del 100% e un F1-score superiore all'82%. Anche le classi *sexism* e *homophobia* ottengono buoni risultati, con valori di F1-score prossimi all'80%.

Tabella 4: Metriche di classificazione per ChatGPT

Classe	Precision	Recall	F1-score	Support
Omofobia	0.739	0.850	0.791	20
Sessismo	0.833	0.750	0.789	20
Razzismo	1.000	0.700	0.824	20
Nessun contenuto offensivo	0.640	0.800	0.711	20
Macro Avg	0.803	0.775	0.779	80
Weighted Avg	0.803	0.775	0.779	80

Matrice di confusione La matrice di confusione mostra che il modello tende a distinguere abbastanza bene tra le classi, pur con alcune sovrapposizioni. I risultati migliori si osservano per la classe *homophobia*, con 17 istanze correttamente classificate su 20, e *no offensive content*, con 16 su 20. Tuttavia, proprio quest'ultima è anche tra le più ambigue: 3 frasi non offensive sono state erroneamente etichettate come *sexism*, e 1 come *homophobia*, suggerendo che ChatGPT adotti un margine prudenziale che può portare a sovrastimare la presenza di contenuti problematici.

La classe *racism* presenta la maggiore dispersione, con 6 frasi su 20 mal classificate, di cui 4 considerate non offensive: ciò potrebbe indicare che alcune espressioni razziste più sottili o indirette non siano colte efficacemente. Al contrario, la classe *sexism* mostra una buona coerenza, con 15 classificazioni corrette e pochi falsi positivi.

Complessivamente, il modello mostra una tendenza a privilegiare il recall nelle classi problematiche, accettando un certo tasso di falsi positivi pur di evitare falsi negativi. Questo comportamento, sebbene non perfetto, è preferibile in contesti sensibili come il rilevamento di discorsi d'odio.

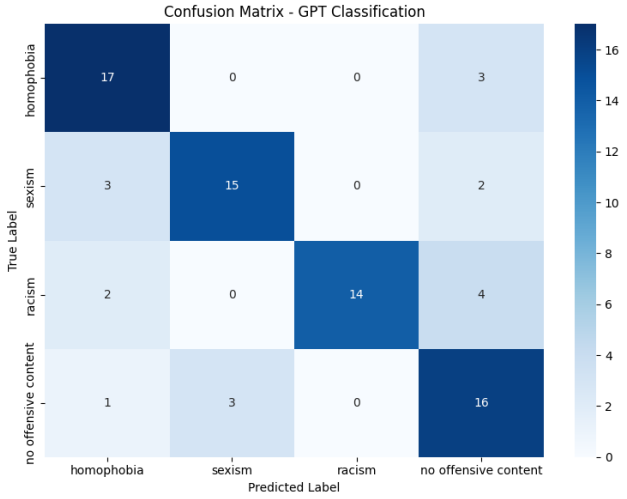


Figura 5: Matrice di confusione la classificazione con ChatGPT

Sintesi e conclusioni sui risultati Va sottolineato che, sebbene il campione sia limitato, il modello riesce a cogliere con discreta efficacia le sfumature offensive, anche senza un training specifico sul nostro dataset. Questo risultato mette in luce il potenziale dei modelli generativi nell'essere utilizzati come classificatori generalisti, riducendo drasticamente la necessità di addestramento supervisionato in contesti simili.

Vantaggi rispetto ai modelli tradizionali L'approccio ha permesso di ottenere un'analisi basata sul significato complessivo della frase, piuttosto che sulla frequenza o combinazione di singoli termini. Questo è un vantaggio notevole rispetto, ad esempio, alla classificazione con TF-IDF + regressione logistica, dove il contesto sintattico viene completamente ignorato, o ai Transformer classici (come BERT), che necessitano di essere fine-tuned su grandi quantità di dati etichettati per ottenere buone prestazioni.

Inoltre, GPT è in grado di cogliere ironia, impliciti culturali e strutture linguistiche complesse, migliorando così la robustezza della classificazione in contesti ambigui o borderline.

Limitazioni L'utilizzo di ChatGPT comporta però alcuni limiti:

- **Costo e dipendenza da API esterne**: l'uso ricorrente del modello via API può risultare oneroso in termini economici e dipende dalla disponibilità di una connessione stabile e di un fornitore di servizi esterno.
- **Non determinismo**: nonostante l'impostazione della temperatura a 0, non è sempre garantita una perfetta riproducibilità dei risultati.
- **Assenza di interpretabilità**: essendo un modello generalista e non addestrato localmente, è difficile spiegare o giustificare in modo preciso le classificazioni prodotte.

4.2.2 Mistral

Il modello `Mistral-7B`, eseguito localmente tramite **Ollama**, è stato utilizzato per classificare il contenuto testuale in quattro categorie: *homophobia*, *sexism*, *racism* e *no offensive content*. I risultati ottenuti evidenziano un livello di accuratezza globale pari al **42%**, con prestazioni inferiori rispetto ai modelli più tradizionali, ma comunque interessanti in un contesto zero-shot e completamente locale, senza addestramento supervisionato.

Tabella 5: Metriche di classificazione per Mistral AI

Classe	Precision	Recall	F1-score	Support
Omofobia	0.381	0.421	0.400	19
Sessismo	0.400	0.400	0.400	20
Razzismo	0.545	0.300	0.387	20
Nessun contenuto offensivo	0.417	0.588	0.488	17
Macro Avg	0.436	0.427	0.419	76
Weighted Avg	0.437	0.421	0.416	76

Dalla Tabella 5 si osserva come il modello tenda ad avere una recall relativamente alta per i casi di contenuto non offensivo, ma prestazioni meno bilanciate nelle classi offensive, in particolare per la categoria *razzismo*, dove la precisione è la più alta tra le classi, ma il recall è la più bassa.

Matrice di confusione Per un'analisi più dettagliata, la Figura 6 mostra la matrice di confusione generata per il modello Mistral, evidenziando le difficoltà nel distinguere tra le classi offensive, spesso confuse tra loro o con la classe neutra.

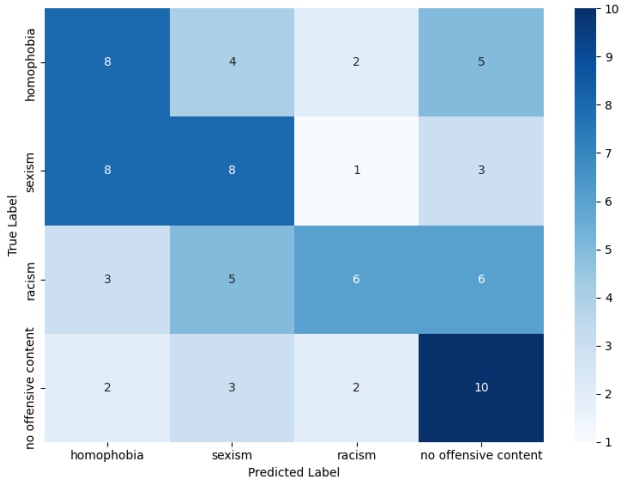


Figura 6: Matrice di confusione per la classificazione con Mistral AI

Sintesi e conclusioni sui risultati Tale risultato può essere attribuito a diversi fattori:

- **Natura zero-shot del modello**: Mistral non è stato fine-tunato sul nostro dataset specifico, né ha ricevuto esempi espliciti di classificazione (few-shot). Questo lo penalizza nella capacità di adattarsi a sfumature linguistiche e contesti particolari.
- **Prompting semplice**: La classificazione è basata su un prompt diretto, senza meccanismi di *chain-of-thought* o rafforzamento del contesto. Modelli LLM spesso richiedono prompt più sofisticati per compiti complessi.
- **Dimensione ridotta del dataset di test**: Il modello è stato valutato su sole 80 frasi, un campione estremamente limitato che non permette di generalizzare le performance né di ridurre l'effetto del rumore statistico. Questa scelta è stata dettata da vincoli pratici: l'inferenza locale su modelli LLM come Mistral, anche in versione quantizzata, richiede tempi considerevoli e risorse computazionali non trascurabili.
- **Ambiguità dei testi**: Molti esempi testuali possono presentare un contenuto ambiguo o borderline tra più classi (ad esempio, frasi sessiste che implicano anche toni omofobi), rendendo difficile per il modello scegliere un'unica etichetta.
- **Limiti del modello quantizzato**: L'utilizzo di una versione quantizzata del modello per favorire la velocità di inferenza su dispositivi locali potrebbe ridurre la capacità del modello di cogliere sfumature semantiche sottili rispetto alla versione a piena precisione.

Questi fattori spiegano la disparità tra le performance osservate e quelle tipicamente attese da un LLM di ultima generazione come Mistral. Nonostante ciò, i risultati confermano la fattibilità dell'uso di modelli open-source in scenari completamente locali, con spazio per miglioramenti tramite tecniche di prompt engineering, few-shot learning o fine-tuning supervisionato.

5 Conclusioni

Il lavoro svolto ha permesso di valutare l'efficacia di differenti approcci di classificazione per l'individuazione di contenuti offensivi di testi e testi trascritti da contenuti multimediali provenienti da piattaforme social. In particolare, sono stati confrontati modelli di machine learning classici (Logistic Regression e SVM lineare), un modello basati su fine-tuning di BERT e soluzioni basate su Large Language Models (LLM), sia in cloud (ChatGPT) che in locale (Mistral).

I modelli di machine learning tradizionali hanno dimostrato ottime performance in termini di accuratezza, con la SVM lineare che ha raggiunto una *accuracy* complessiva del **94.91%**. Questi approcci, sebbene robusti e interpretabili, presentano tuttavia una limitazione strutturale: l'incapacità di cogliere il contesto globale della frase. L'uso di rappresentazioni *bag-of-words* come TF-IDF, seppur efficace per pattern lessicali evidenti, non consente di comprendere ironia, ambiguità sintattiche o impliciti semantici, elementi spesso centrali nella rilevazione del linguaggio offensivo.

Il fine-tuning di BERT ha portato a un notevole incremento delle performance, con un'accuratezza del 98.59% e metriche F1 molto elevate anche per la classe neutra. Tuttavia, l'analisi delle curve di loss ha evidenziato un chiaro fenomeno di *overfitting*: mentre la *training loss* si stabilizza su valori molto bassi, la *validation loss* cresce progressivamente nelle ultime epoche. Questo comportamento suggerisce che il modello stia memorizzando troppo fedelmente i dati di addestramento, perdendo capacità di generalizzazione. Le cause principali possono essere ricondotte alla ridotta dimensione del dataset rispetto alla complessità del modello e alla mancanza di tecniche di regolarizzazione più avanzate (es. early stopping, dropout mirato o data augmentation testuale).

I LLM hanno mostrato risultati eterogenei ma promettenti. In particolare, `ChatGPT`, nonostante sia stato utilizzato in modalità *zero-shot* e senza alcun fine-tuning sul dataset specifico, ha raggiunto un'accuratezza del **77.5%**, con buone prestazioni in termini di F1-score nelle classi offensive. Questo evidenzia come i modelli generativi generalisti siano in grado di cogliere sfumature linguistiche, impliciti semantici e strutture complesse, senza necessità di addestramento supervisionato. La capacità di ChatGPT di integrare contesto, tono e intenzione comunicativa costituisce un vantaggio significativo rispetto a modelli basati esclusivamente su feature lessicali o sintattiche. Al contrario, il modello `Mistral-7B`, eseguito in locale in condizioni analoghe (zero-shot, senza fine-tuning), ha ottenuto un'accuratezza del **42%**. Sebbene le sue performance siano inferiori, il risultato dimostra la fattibilità di un approccio completamente locale alla classificazione di contenuti offensivi, con margini di miglioramento attraverso tecniche di prompt engineering, few-shot learning o fine-tuning supervisionato. È importante notare che Mistral è stato testato su un campione limitato e in versione quantizzata, fattori che possono aver inciso sulla qualità dell'output.

Questi risultati suggeriscono che un approccio ibrido che combini un primo filtro leggero basato su modelli classici con una successiva analisi semantica tramite LLM potrebbe rappresentare la soluzione più efficace, bilanciando accuratezza, scalabilità e comprensione linguistica.

5.1 Sviluppi futuri

Alla luce dei risultati ottenuti e delle limitazioni riscontrate, sono diversi gli ambiti nei quali il sistema proposto può essere ulteriormente sviluppato e migliorato:

- **Espansione del dataset**: L'overfitting osservato durante il fine-tuning di BERT evidenzia la necessità di disporre di un volume maggiore di dati etichettati. Una possibile direzione è l'ampliamento del dataset tramite *data augmentation* semantico oppure l'acquisizione di nuovi dati reali.

- **Utilizzo di tecniche di regolarizzazione avanzate**: Per migliorare la generalizzazione del modello BERT, sarebbe utile integrare strategie come *early stopping*, *dropout adattivo* o *learning rate scheduling* che consentano di ridurre l'overfitting senza penalizzare l'efficacia predittiva.

- **Integrazione con modelli LLM fine-tuned localmente**: Una futura evoluzione potrebbe consistere nel fine-tuning locale di modelli come Mistral o LLaMA, sfruttando dataset personalizzati per ottenere un comportamento più preciso e controllabile, mantenendo i vantaggi in termini di privacy.

- **Classificazione multi-label**: Il sistema attuale classifica il contenuto offensivo in un'unica categoria, ma nella realtà un singolo messaggio può contenere più forme di discriminazione simultaneamente (es. sessismo e omofobia). L'adozione di un paradigma *multi-label* permetterebbe di etichettare correttamente tali casi complessi, riflettendo meglio la natura sfumata del linguaggio offensivo. La mancata considerazione di questa possibilità può aver contribuito, almeno in parte, all'abbassamento dell'accuracy nei modelli testati, poiché frasi con offese multiple venivano forzatamente assegnate a una sola categoria, inducendo errori anche nei modelli più sofisticati.

- **Rilevamento del tono e dell'ironia**: L'identificazione di contenuti offensivi può essere complicata dalla presenza di sarcasmo, ironia o ambiguità retoriche. L'integrazione con mo-

delli specializzati nel riconoscimento del tono comunicativo potrebbe migliorare la sensibilità del sistema in contesti borderline.

- **Deploy di un'applicazione interattiva**: Infine, l'integrazione del sistema in una piattaforma web accessibile permetterebbe l'utilizzo in scenari reali, ad esempio per moderatori di contenuti social o strumenti di auto-monitoraggio per creator digitali.

Nel complesso, questi sviluppi mirano non solo a raffinare le capacità tecniche del sistema, ma anche a renderlo più etico, scalabile e orientato all'uso in contesti reali, dove la rilevazione automatica del linguaggio offensivo rappresenta una sfida in continua evoluzione.

Riferimenti bibliografici

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2019.

[2] TikTok, "Tiktok — make your day," https://www.tiktok.com/, 2025, [Online; accessed 11-July-2025].

[3] Threads, "Threads by meta," https://www.threads.net/, 2025, [Online; accessed 11-July-2025].

[4] Hugging Face, "Hugging face datasets," https://huggingface.co/datasets, 2025, accessed: 11 July 2025.

[5] JoshMcGiff, "Homophobia detection twitterx," 2020. [Online]. Available: https://huggingface.co/datasets/JoshMcGiff/HomophobiaDetectionTwitterX/tree/main

[6] Kaggle, "Kaggle: Your machine learning and data science community," https://www.kaggle.com, 2025, accessed: 11 July 2025.

[7] Aadyasingh55, "Sexism detection in english texts," 2021. [Online]. Available: https://www.kaggle.com/datasets/aadyasingh55/sexism-detection-in-english-texts?resource=download

[8] R. Abdullah, "Racism tweets," 2021. [Online]. Available: https://www.kaggle.com/datasets/raghadabdullah/racism-tweets

[9] OpenAI, "ChatGPT: Optimizing Language Models for Dialogue," https://openai.com/chatgpt, 2023, accessed: 2025-07-11.

[10] D. Team and U. AI, "Toxic bert - a bert model fine-tuned for toxic comment classification," https://huggingface.co/unitary/toxic-bert, 2020, accessed: 2025-07-12.

[11] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed. John Wiley & Sons, 2013.

[12] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Transformers: State-of-the-art natural language processing," *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.6

[15] M. AI, "Mistral 7b - open-weight language model," https://huggingface.co/mistralai/Mistral-7B-v0.1, 2023, accessed: 2025-07-12.

[16] G. Gerganov and llama.cpp team, "Gguf: Gpt-generated unified format," https://github.com/ggerganov/llama.cpp, 2023, format introduced in llama.cpp for efficient quantized LLM inference; accessed: 2025-07-14.

[17] O. Team, "Ollama: Run large language models locally," https://ollama.com, 2024, accessed: 2025-07-12.

[18] Selenium contributors, "Selenium webdriver," https://www.selenium.dev/, 2025, [Online; accessed 11-July-2025].

[19] L. Richardson, "Beautiful soup documentation," https://www.crummy.com/software/BeautifulSoup/, 2023, [Online; accessed 11-July-2025].

[20] OpenAI, "Whisper: Robust speech recognition via large-scale weak supervision," https://github.com/openai/whisper, 2023, [Online; accessed 11-July-2025].