

UNIVERSITÀ DI PISA

Dipartimento di Filologia, Letteratura e Linguistica

Laurea Magistrale in Informatica Umanistica

## Linguistica Computazionale II

Cross-Genre Gender Prediction in Italian

Di Prizio Emily  
Mat. 684472

---

Anno Accademico 2023/2024

# Contents

<b>Introduzione</b>	<b>1</b>
0.1 Preparazione dei dati . . . . .	1
<b>1 Classificatore SVM con caratteristiche linguistiche non lessicali</b>	<b>2</b>
1.1 Caricamento del dataset, train-test splitting e 5-fold validation . . . . .	2
1.2 Test del modello sul test set ufficiale del task . . . . .	3
1.3 Feature Importance . . . . .	3
<b>2 Classificatore SVM con n-grammi di caratteri, parole e POS</b>	<b>4</b>
2.1 Caricamento del dataset, estrazione n-grammi e estrazione delle feature . . . . .	4
2.2 Train-test splitting, normalizzazione e 5-fold validation . . . . .	4
2.3 Test del modello sul test set ufficiale del task . . . . .	5
<b>3 Classificatore SVM con word embeddings</b>	<b>7</b>
3.1 Pre-processing . . . . .	7
3.2 Estrazione delle feature . . . . .	7
3.3 Train-test splitting, normalizzazione e 5-fold validation . . . . .	7
3.4 Test del modello sul test set ufficiale del task . . . . .	8
<b>4 Neural Language Model</b>	<b>9</b>
4.1 Pre-processing . . . . .	9
4.2 Addestramento del modello . . . . .	9
4.3 Valutazione del modello sul test set ufficiale del task . . . . .	10
<b>5 Conclusioni</b>	<b>11</b>

# Introduzione

Il progetto si concentra sul task di Cross-Genre Gender Prediction proposto nell’ambito di EVALITA 2018, il cui obiettivo è predire il genere dell’autore di un testo (maschile o femminile) utilizzando testi provenienti da vari generi, come articoli di giornale, post sui social media e altri contenuti scritti. Questo compito mira ad analizzare come le caratteristiche linguistiche possano rivelare informazioni sul genere dell’autore, considerando le variazioni stilistiche in base al genere testuale.

Per realizzare il classificatore, saranno sviluppati diversi modelli basati su differenti rappresentazioni del testo. Il primo modello prevede l’utilizzo di un classificatore SVM lineare che impiega una rappresentazione del testo basata su informazioni linguistiche non lessicali estratte tramite il sistema Profiling-UD. Il secondo modello, anch’esso basato su un SVM lineare, utilizza una rappresentazione del testo basata su n-grammi di caratteri, parole e part-of-speech, esplorando diverse varianti relative alla lunghezza degli n-grammi e al tipo di informazione (forme, lemmi, caratteri, part-of-speech). Il terzo modello prevede un classificatore SVM lineare che si avvale di una rappresentazione del testo costruita attraverso l’uso di word embeddings, testando diverse combinazioni degli embedding delle parole e delle categorie grammaticali.

Inoltre, verrà esplorato un ulteriore approccio utilizzando un neural language model, sul quale verrà condotto un processo di fine-tuning per cinque epoch. Questo approccio sfrutterà le potenzialità dei modelli di linguaggio neurale per catturare informazioni più complesse e profonde, con l’obiettivo di migliorare ulteriormente le performance del classificatore.

## 0.1 Preparazione dei dati

Per iniziare, ho scaricato i dataset di training e test - già separati - da GitHub, che contengono cinque file in formato .txt appartenenti a diversi generi: children, diary, journalism, twitter, e youtube. Tuttavia, nel dataset di test, la colonna del genere (gender) conteneva un punto interrogativo al posto delle etichette “F” o “M”. Inoltre, era presente un altro file denominato GOLD, che associava a ciascun documento un ID e il genere corrispondente. Ho quindi effettuato una mappatura tra i generi presenti nei file GOLD e gli ID nei documenti di test per correggere questa incongruenza e associare i generi ai rispettivi documenti.

Durante la fase di pre-processing, ho estratto le informazioni dall’header di ciascun documento presente nei file di training e test, tra cui l’ID del documento, il genere e il sesso dell’autore. Successivamente, ho creato i percorsi dei file di output per Profiling-UD, denominando ciascun file con una struttura del tipo `training#ID#genre#gender.txt`. In questo modo, ogni documento è stato trattato separatamente e i contenuti sono stati salvati in nuovi file. Inoltre, durante il processo, ho verificato e rimosso eventuali file contenenti caratteri non leggibili o terminatori di riga incompatibili (come quelli in formato CRLF o con codifica non UTF-8), per garantire che i dati fossero compatibili con Profiling-UD. Una volta completata questa fase di pulizia, i documenti sono stati passati a Profiling-UD. I risultati delle analisi sono stati salvati in un file CSV. La prima riga del file contiene l’intestazione con i nomi delle feature, tra cui: `Filename, n_sentences, n_tokens, tokens_per_sent, ..., subordinate_dist_5`. Ogni riga successiva rappresenta l’analisi di un singolo documento: il primo campo indica il nome del file analizzato, mentre i successivi contengono i valori delle feature estratte, utilizzabili per i successivi modelli di classificazione.

Per quanto riguarda i classificatori, il task scelto è un modello cross-genre non-Diaries per il genere Diaries.

# 1 Classificatore SVM con caratteristiche linguistiche non lessicali

Il primo classificatore si basa sull'uso di caratteristiche linguistiche non lessicali. Questo significa che il modello non apprende dai contenuti semantici o dalle parole chiave, ma riconosce schemi stilistici e strutturali nei testi. La classificazione, quindi, non dipende da cosa viene detto, ma da come viene espresso.

## 1.1 Caricamento del dataset, train-test splitting e 5-fold validation

Per costruire il dataset, ogni documento è stato rappresentato come un insieme di feature estratte attraverso Profiling-UD. Per ottimizzare il flusso di lavoro, sono state implementate diverse funzioni ausiliarie nel file functions.py, utilizzate in più classificatori, permettono di gestire in modo efficiente le diverse fasi della pipeline di classificazione. Una delle operazioni fondamentali è la suddivisione del dataset in training set e test set, che segue una logica di cross-genre. Il training set include documenti appartenenti a tutti i generi disponibili tranne il genere Diaries, mentre il test set è composto esclusivamente da documenti appartenenti a questo genere. Questa scelta permette di valutare la capacità del modello di generalizzare su un genere mai visto in fase di addestramento.

Dopo un processo di normalizzazione, per valutare la robustezza del classificatore, è stata implementata una validazione incrociata con 5-fold (5-fold cross-validation). In questo modo, si ottiene una valutazione più affidabile delle prestazioni del modello, riducendo la dipendenza dalla specifica suddivisione iniziale dei dati. Per stabilire una baseline di riferimento, è stato inoltre utilizzato un Dummy Classifier, che effettua previsioni casuali o basate su semplici euristiche, come la classificazione sempre nella classe più frequente. Questo test permette di confrontare le prestazioni dell'SVM con un modello di riferimento minimale, verificando se il classificatore sfrutta realmente le informazioni strutturali presenti nei dati.

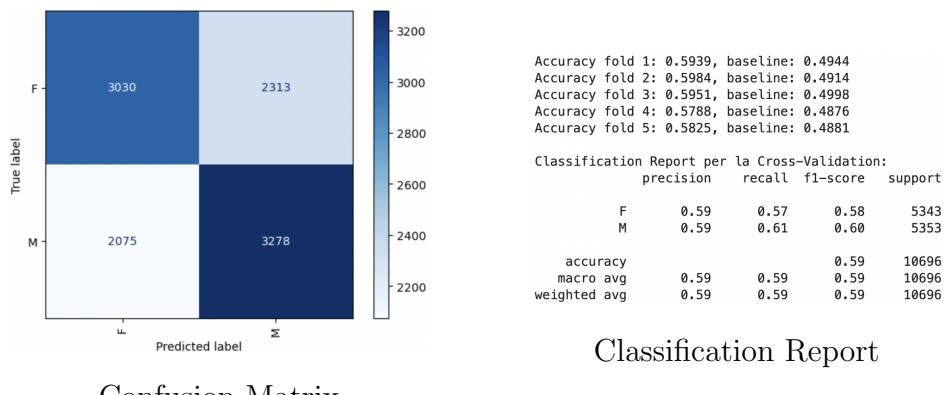


Figure 1.1: Confusion Matrix e Classification Report

Come possiamo vedere dai risultati riportati, vedi Figura 4.2, l'accuracy varia leggermente tra le fold, con valori compresi tra 0.5788 e 0.5984. Questo indica che il modello ha una performance coerente e non è soggetto a variazioni eccessive nei diversi sottoinsiemi del dataset. La baseline accuracy, ottenuta con un Dummy Classifier che probabilmente fa previsioni casuali o basate sulle distribuzioni di classe, è inferiore (circa 0.49). Questo

significa che il modello basato su caratteristiche linguistiche non lessicali ha una capacità predittiva migliore del caso. Il classification report mostra che le metriche di valutazione (precision, recall, f1-score) sono quasi identiche per le due classi (“F” e “M”), con valori attorno a 0.59. Questo indica che il modello non è sbilanciato a favore di una classe specifica. I risultati, quindi, mostrano che esistono pattern stilistici legati al genere dell’autore, ma non abbastanza distintivi per una classificazione accurata. La bassa differenza rispetto alla baseline suggerisce che lo stile linguistico non sia fortemente influenzato dal genere nei testi analizzati.

## 1.2 Test del modello sul test set ufficiale del task

Il modello è stato testato sul test set ufficiale del task per valutare le sue prestazioni su dati non visti.

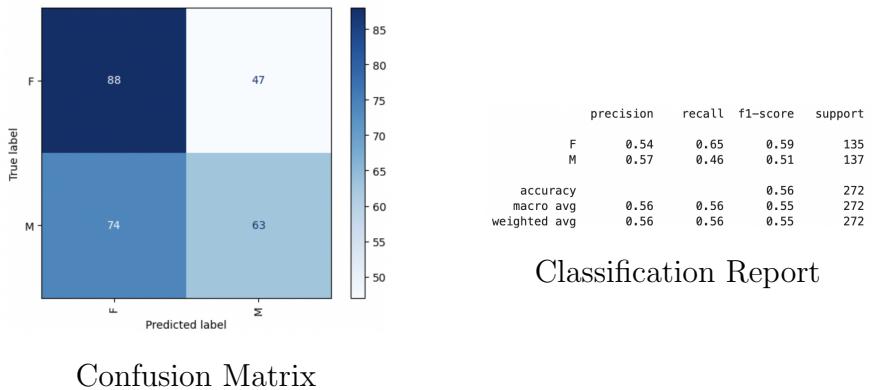


Figure 1.2: Confusion Matrix e Classification Report sul test set ufficiale

I risultati nella figura 4.2 mostrano un’accuratezza del 56%, inferiore rispetto alla validazione incrociata. La confusion matrix indica una maggiore capacità nel riconoscere la classe “F” rispetto alla classe “M”, con un recall più alto (0.65 vs 0.46). Tuttavia, il modello fatica a distinguere correttamente gli esempi maschili, con numerosi errori di classificazione. Il f1-score riflette questo squilibrio, suggerendo che le feature non lessicali potrebbero non essere sufficienti per una distinzione accurata in questo contesto.

## 1.3 Feature Importance

L’analisi delle 15 feature più importanti mostra che la variabile `tokens_per_sent` ha il peso maggiore nella classificazione. Questo significa che la lunghezza media delle frasi è un elemento chiave nel processo decisionale del modello. Un numero elevato di token per frase può indicare una maggiore complessità sintattica e strutturale, influenzando la distinzione tra le classi. Frasi più lunghe potrebbero contenere strutture linguistiche più articolate, subordinate o un uso più frequente di connettivi, elementi che il modello ha identificato come discriminanti principali.

Inoltre, altre feature come la lunghezza media delle parole e il numero di subordinate per frase giocano un ruolo chiave nella classificazione. Queste caratteristiche indicano una variazione nello stile di scrittura che il modello riesce a riconoscere e sfruttare per la predizione del genere dell’autore.

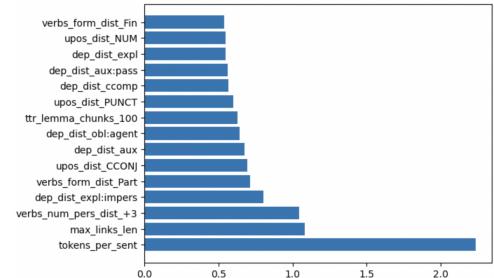


Figure 1.3: Importanza delle feature nel modello di classificazione

## 2 Classificatore SVM con n-grammi di caratteri, parole e POS

Il secondo classificatore utilizza n-grammi di caratteri, parole, lemmi e part-of-speech per rappresentare il testo. Questo approccio cattura sia informazioni lessicali che strutturali, permettendo al modello di riconoscere pattern stilistici distintivi tra autori di genere diverso.

### 2.1 Caricamento del dataset, estrazione n-grammi e estrazione delle feature

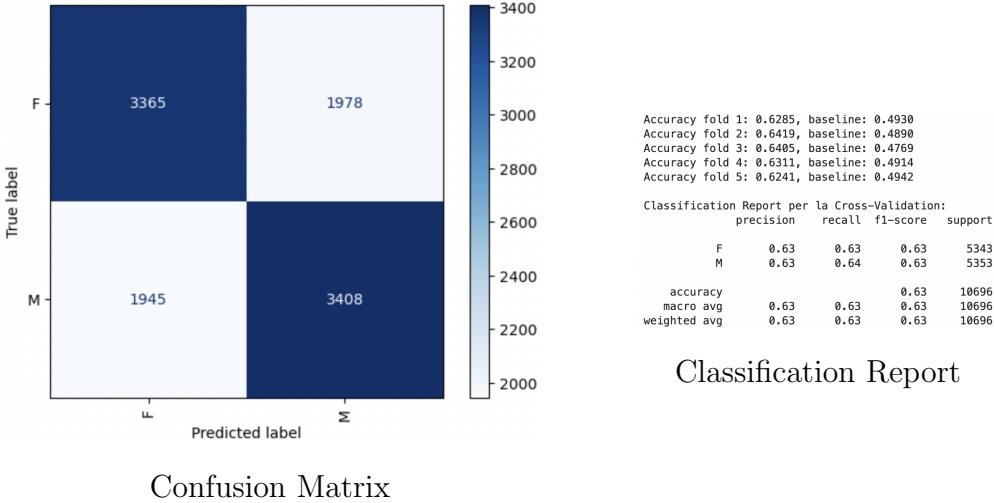
In questa fase, il dataset viene caricato e preprocessato per estrarre n-grammi di caratteri, parole, lemmi e part-of-speech, che rappresentano schemi linguistici utili per la classificazione. Gli n-grammi permettono di catturare sia informazioni locali, come combinazioni di lettere o parole frequenti, sia strutture sintattiche più ampie attraverso sequenze di POS. Per l'estrazione, si utilizzano due funzioni distinte: una dedicata agli n-grammi basati su parole, lemmi e POS, e un'altra per gli n-grammi di caratteri. Si scelgono uni-, bi- e trigrammi per ciascuna categoria, in modo da garantire una rappresentazione ricca del testo. Successivamente, le feature estratte vengono normalizzate e organizzate in base alla tipologia di n-gramma o alla categoria linguistica. In particolare, vengono salvate nelle seguenti posizioni della lista `self.feat_ngr`: [0]: tutti gli n-grammi; [1]: n-grammi basati su parole; [2]: n-grammi di caratteri; [3]: unigrammi di tutte le categorie; [4]: trigrammi di tutte le categorie. Grazie a questa struttura, è possibile combinare diverse configurazioni di n-grammi e memorizzarle in nuove posizioni della lista, consentendo di testare varie strategie di rappresentazione per ottimizzare il modello di classificazione.

### 2.2 Train-test splitting, normalizzazione e 5-fold validation

Utilizzando tre funzioni definite in `functions.py`, è stato effettuato il train-test splitting. Successivamente, tramite la funzione `process_features`, sono state eseguite tre operazioni fondamentali: il filtro delle feature poco frequenti, la creazione del vettore delle feature e la loro normalizzazione. Infine, è stata eseguita la 5-fold cross-validation e il test con una baseline. Questi passaggi sono stati eseguiti automaticamente tramite le funzioni in `functions.py`, garantendo un processo strutturato e riproducibile per ogni gruppo di feature precedentemente definito.

L'analisi della k-fold cross-validation conferma che l'uso di tutti gli n-grammi (4.2) è il più efficace, combinando informazioni lessicali, strutturali e morfosintattiche per massimizzare precisione e richiamo.

L'insieme di tutti gli unigrammi (2.3a), che includono unigrammi di parole, caratteri, lemma e POS, sono già forti predittori, ma l'aggiunta di bigrammi e trigrammi migliora ulteriormente le prestazioni. Gli n-grammi di parole (2.2a) soffrono l'aumento della dimensionalità, mentre quelli di caratteri (2.2b) catturano pattern locali ma senza semantica. I soli trigrammi (2.3b), invece, risultano meno flessibili. In sintesi, la combinazione di più livelli di granularità e informazione linguistica è la chiave per una classificazione ottimale.



Confusion Matrix

Figure 2.1: Confusion Matrix e Classification Report: tutti gli ngrammi

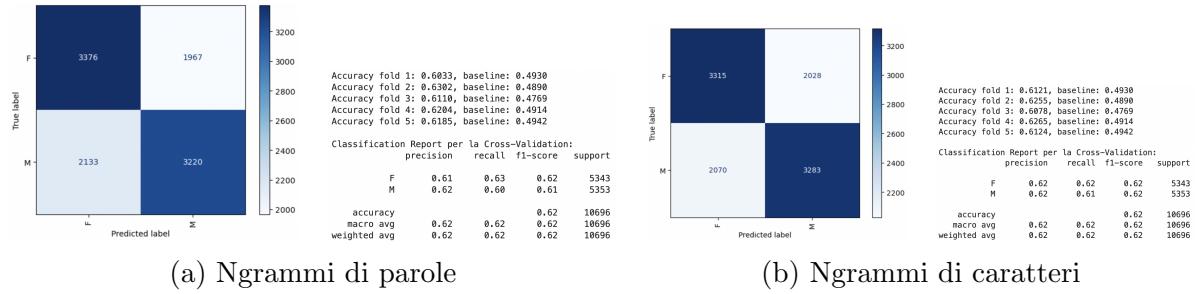


Figure 2.2: Confronto tra ngrammi di parole e ngrammi di caratteri

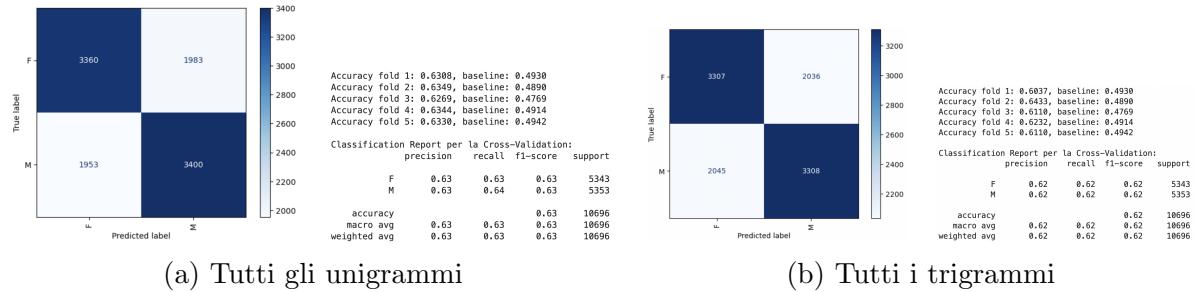


Figure 2.3: Confronto tra tutti gli unigrammi e tutti i trigrammi

## 2.3 Test del modello sul test set ufficiale del task

Il test finale è stato condotto sul test set ufficiale per valutare le sue prestazioni su dati non visti utilizzando il gruppo di feature più performante emerso dalla 5-fold validation, ovvero tutti gli n-grammi. I risultati sul test set ufficiale confermano l'efficacia dell'approccio basato su tutti gli n-grammi, che mantiene alte prestazioni in termini di accuratezza e bilanciamento tra precision e recall. La matrice di confusione mostra una buona capacità del modello nel distinguere la maggior parte delle classi.

Il classification report evidenzia che le metriche rimangono in linea con quelle della validazione, confermando una buona generalizzazione del modello.uttavia, alcune classi pre-

sentano variazioni nei valori di precision e recall, suggerendo che fattori come la complessità linguistica o la somiglianza tra alcune categorie possano aver influito sulle predizioni.

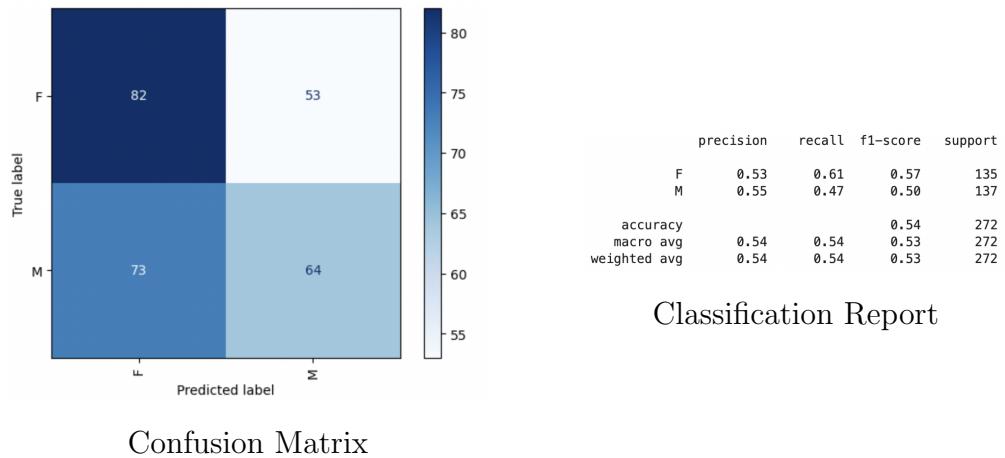


Figure 2.4: Confusion Matrix e Classification Report sul test set ufficiale

# 3 Classificatore SVM con word embeddings

Il terzo classificatore utilizza word embeddings per rappresentare il testo, catturando relazioni semantiche tra le parole. Questo approccio consente a SVM di sfruttare informazioni più profonde sul significato e lo stile del linguaggio.

## 3.1 Pre-processing

Il preprocessing inizia con il download degli embedding pre-addestrati nella versione ItWaC 128. Per garantirne la compatibilità con l'implementazione successiva, gli embedding vengono convertiti in formato .txt, semplificando la gestione e il caricamento nel modello. Successivamente, viene caricato il dataset ottenuto dall'analisi con Profiling UD e da esso vengono estratti i token, che rappresentano le unità di analisi del modello. Dopo l'estrazione dei token, viene applicata una normalizzazione del testo. I numeri vengono sostituiti in base alla loro lunghezza o convertiti in un simbolo generico (@Dg), mentre gli URL e le parole eccessivamente lunghe vengono rimpiazzati con segnaposti. Inoltre, la capitalizzazione viene adattata per uniformare la rappresentazione delle parole. Una funzione dedicata estrae i token dai file generati da Profiling UD, creando per ciascuno un dizionario con la parola normalizzata e il relativo POS. I token composti vengono gestiti correttamente, e la normalizzazione del testo assicura coerenza nei dati.

## 3.2 Estrazione delle feature

Per rappresentare ogni documento, gli embedding delle parole vengono aggregati tramite diverse funzioni. I metodi di aggregazione utilizzati sono: prodotto degli embedding delle parole piene; prodotto degli embedding divisi per POS e concatenati; media degli embedding delle parole piene; media degli embedding divisi per POS e concatenati. Infine, viene costruita la matrice delle feature per il training set per ogni combinazione definita sopra.

## 3.3 Train-test splitting, normalizzazione e 5-fold validation

Utilizzando le funzioni definite in functions.py, è stato effettuato il train-test splitting del dataset. Successivamente, le feature sono state normalizzate per garantire una corretta comparabilità tra i valori. Infine, è stata eseguita una 5-fold validation su ciascuno dei quattro gruppi di embedding definiti in precedenza, valutando le loro performance nel classificatore.

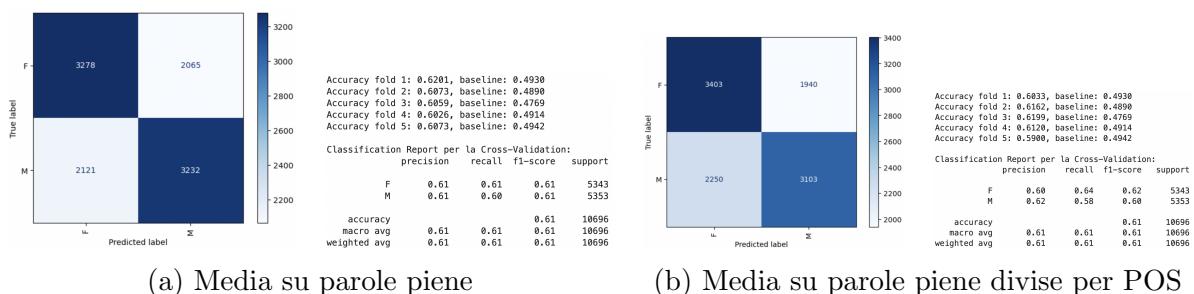


Figure 3.1: Confronto tra media su parole piene e media su parole piene divise per POS

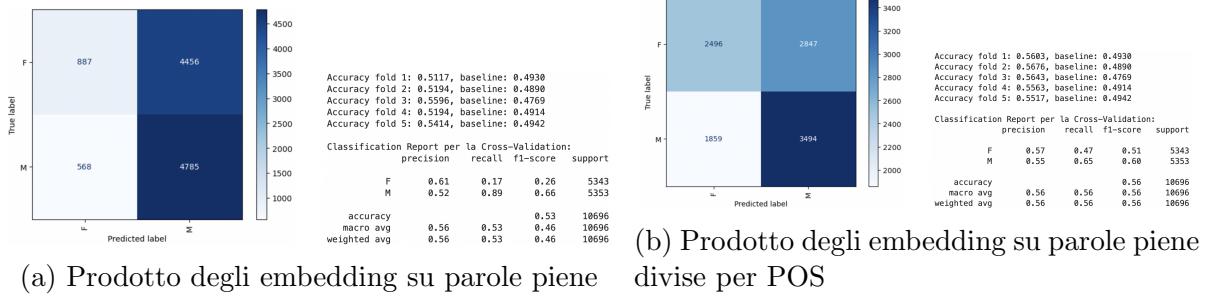


Figure 3.2: Confronto tra prodotto degli embedding su parole piene e prodotto degli embedding su parole piene divise per POS

L’aggregazione tramite media degli embedding delle parole piene (Figura 3.1a) si dimostra l’approccio più efficace, garantendo metriche complessivamente bilanciate tra le classi e una buona capacità di generalizzazione. Separare gli embedding per POS prima di calcolarne la media (Figura 3.1b) non porta a miglioramenti significativi e, anzi, si osserva un comportamento differenziato tra le classi. In particolare, le metriche migliorano leggermente per la classe “F”, suggerendo che la separazione per categorie grammaticali possa evidenziare alcune caratteristiche distintive di questa classe. Tuttavia, per la classe “M”, si nota un lieve peggioramento delle prestazioni.

L’aggregazione tramite prodotto degli embedding delle parole piene (Figura 3.2a) mostra risultati nettamente peggiori rispetto alla media. Questo risultato si riflette anche nei valori di F1-score, più bassi rispetto agli approcci basati sulla media, e in una distribuzione degli errori meno bilanciata. La separazione per POS (Figura 3.2b) non migliora significativamente le prestazioni, suggerendo che il prodotto potrebbe non essere un’operazione adeguata per aggregare vettori di embedding in questo contesto.

### 3.4 Test del modello sul test set ufficiale del task

Per la valutazione finale, il modello è stato testato sul test set ufficiale, utilizzando il gruppo di feature più performante emerso dalla 5-fold validation, ovvero la media degli embedding delle parole piene. Il modello mostra una maggiore capacità di riconoscere testi scritti da autrici, con un recall più alto per la classe “F” rispetto alla classe “M”. Tuttavia, la precisione è superiore per la classe “M”, indicando una tendenza a classificare erroneamente testi maschili come femminili. La matrice di confusione evidenzia infatti un numero significativo di errori in questa direzione, mentre gli errori inversi sono meno frequenti. Complessivamente, l’accuratezza è moderata, suggerendo margini di miglioramento nella distinzione tra le due classi.

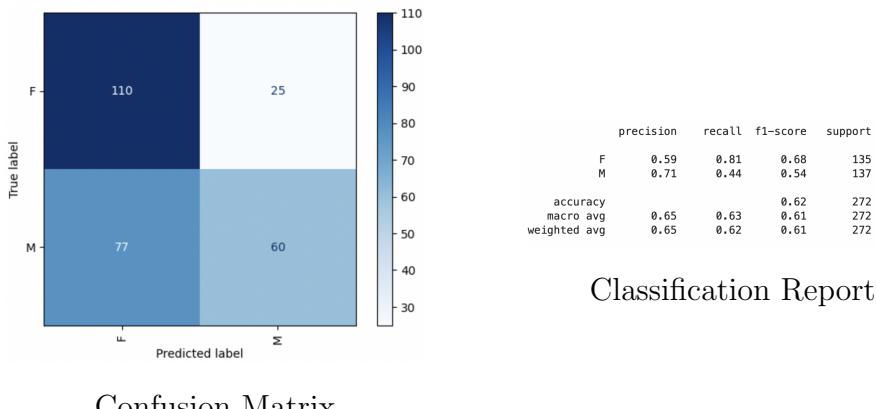


Figure 3.3: Confusion Matrix e Classification Report sul test set ufficiale

# 4 Neural Language Model

Per l'ultimo classificatore si utilizza un modello di linguaggio neurale basato su BERT, che sfrutta un'architettura bidirezionale per catturare il contesto delle parole e migliorare la classificazione.

## 4.1 Pre-processing

In questa fase, si procede con il caricamento dei dati, che vengono poi convertiti nel formato datasets per facilitarne la gestione. Successivamente, il dataset di training viene suddiviso in training e validation set per garantire una corretta valutazione del modello. Una volta preparati i dati, si carica il modello bert-base-italian-cased, una variante di BERT specifica per la lingua italiana. Infine, si procede con la tokenizzazione del testo.

## 4.2 Addestramento del modello

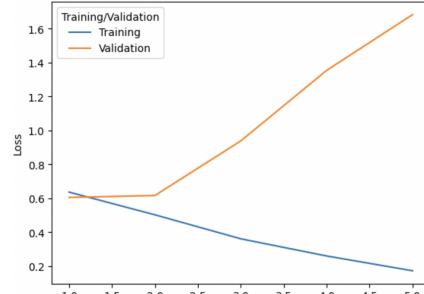
Dopo la fase di pre-processing, il modello viene addestrato attraverso un processo di fine-tuning su **5 epoch**. L'addestramento viene gestito tramite il framework *Transformers*, definendo:

- I parametri di training, tra cui il numero di epoch, il learning rate, la dimensione del batch e le strategie di salvataggio e logging.
- Una metrica di valutazione per monitorare le performance del modello durante il processo.
- Il trainer, responsabile dell'ottimizzazione del modello e dell'esecuzione del processo di addestramento.

L'obiettivo del fine-tuning è adattare il modello pre-addestrato ai dati specifici del task di classificazione, migliorandone la capacità di generalizzazione.

Epoch	Training Loss	Validation Loss	F1
1	0.636800	0.605541	0.667307
2	0.503100	0.616850	0.687604
3	0.361800	0.938317	0.684119
4	0.260700	1.353923	0.679257
5	0.173600	1.682029	0.673821

Loss e f1-score



training e validation loss

Figure 4.1: Evoluzione delle metriche di addestramento durante il fine-tuning

Durante il fine-tuning su cinque epoch, la training loss diminuisce costantemente, indicando che il modello sta imparando efficacemente dai dati di addestramento. Tuttavia, la validation loss rimane stabile nelle prime due epoch, per poi aumentare drasticamente dalla terza in poi. Questo è un chiaro segnale di overfitting: il modello si adatta eccessivamente ai dati di training, perdendo capacità di generalizzazione sui dati di validazione. Di conseguenza, il punteggio F1 raggiunge un valore massimo alla seconda epoca, per poi calare progressivamente, confermando il deterioramento delle prestazioni del modello

sulle nuove istanze. Un possibile miglioramento sarebbe stato l'uso di un meccanismo di early stopping, interrompendo l'addestramento alla seconda o terza epoca per evitare l'overfitting.

### 4.3 Valutazione del modello sul test set ufficiale del task

Dopo l'addestramento del modello, viene eseguita una valutazione sulle prestazioni utilizzando il test set ufficiale.

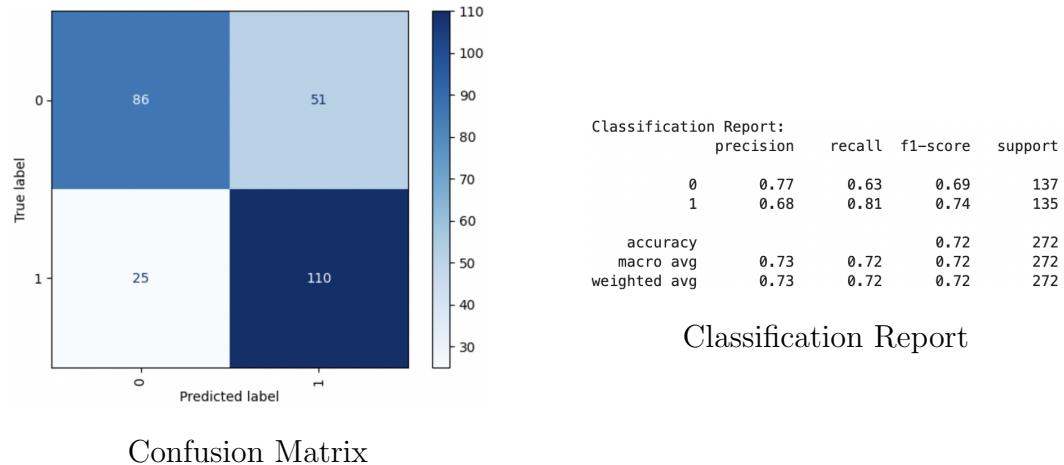


Figure 4.2: Confusion Matrix e Classification Report sul test set ufficiale

Il modello mostra una tendenza a riconoscere meglio la classe “M” rispetto alla classe “F”. In particolare, è più propenso a classificare correttamente i campioni appartenenti a “M”, mentre ha maggiori difficoltà nel distinguere correttamente quelli di “F”, portando a un numero più elevato di errori su questa categoria.

Questo comportamento suggerisce che il modello potrebbe aver appreso caratteristiche più distintive per “M”, mentre quelle relative a “F” risultano più difficili da identificare con precisione.

## 5 Conclusioni

In questo studio, sono stati sperimentati diversi approcci di classificazione per il task di Cross-Genre Gender Prediction, valutando l'efficacia di SVM con caratteristiche linguistiche non lessicali, SVM con n-grammi, SVM con word embeddings e un modello neurale basato su BERT.

I risultati hanno mostrato che il miglior approccio è stato il Neural Language Model (BERT), che ha raggiunto le performance più elevate in termini di accuratezza e capacità di generalizzazione. Questo risultato era atteso, poiché i modelli di linguaggio neurali sono progettati per catturare relazioni complesse nel testo, sfruttando rappresentazioni distribuzionali e il fine-tuning su dati specifici. Tuttavia, l'analisi ha evidenziato problemi di overfitting dopo la seconda epoca, suggerendo che un'ottimizzazione dei parametri, come l'early stopping, potrebbe migliorare ulteriormente le prestazioni.

Il classificatore basato su n-grammi si è dimostrato una valida alternativa, con buone prestazioni nella distinzione tra i generi. La combinazione di tutti gli n-grammi ha garantito il miglior compromesso tra precisione e richiamo, mostrando che la struttura lessicale e sintattica dei testi contiene informazioni rilevanti per la classificazione del genere.

L'uso di word embeddings ha prodotto risultati più contrastanti: la media degli embedding delle parole piene ha portato a risultati discreti, mentre l'uso del prodotto degli embedding ha mostrato prestazioni inferiori, indicando che non tutte le strategie di aggregazione sono efficaci per questo tipo di task.

Infine, il classificatore SVM con caratteristiche linguistiche non lessicali ha ottenuto le performance meno competitive, confermando che la sola analisi stilistica e strutturale del testo non è sufficiente a distinguere efficacemente il genere dell'autore.

In conclusione, il modello neurale rappresenta la scelta più efficace per il task, ma con possibili margini di miglioramento nella gestione dell'overfitting. I risultati suggeriscono che combinare diverse strategie, come il fine-tuning dei modelli neurali con feature linguistiche specifiche, potrebbe portare a un ulteriore incremento delle prestazioni.