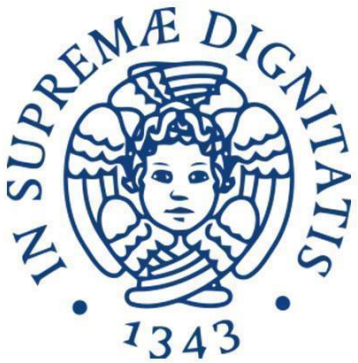

UNIVERSITÀ DI PISA

INFORMATICA UMANISTICA

LINGUISTICA COMPUTAZIONALE II

IronITA – Irony Detection in Twitter

Chiara Giurdanella (560686)

ANNO ACCADEMICO 2022/2023

Indice

1. Introduzione

Il progetto è incentrato su un task di classificazione proposto da EVALITA[1] nel 2018.
Nello specifico, si è scelto di affrontare il sub-task A, relativo alla classificazione di un tweet come ironico o non ironico.
Dopo una fase di presentazione del set di dati impiegato, le sezioni che seguono sono incentrate sullo sviluppo di tre classificatori *SVM* lineari, che prendono in input differenti rappresentazioni del testo, e sull'implementazione di un *Neural Language Model*.

2. Dataset

Il dataset di IronITA[2] (Irony Detection in Italian Tweets) include tweet su questioni sociali, come l'incitamento all'odio e la presa di posizione politica, testi in cui è particolarmente interessante studiare l'uso e il ruolo di espedienti figurativi come l'ironia e il sarcasmo. Il set di dati si compone di 4849 righe e 6 colonne, la cui semantica è mostrata nella Tabella 1.

Attributo	Data type	Descrizione	Range di valori
twitter_id	String	Id assegnato a un tweet	4849 valori distinti
text	String	Testo del tweet	
irony	String	Indica se il tweet è ironico o non ironico	Due valori distinti: 0 e 1
sarcasm	String	Indica se il tweet è sarcastico o non sarcastico	Due valori distinti: 0 e 1
source	String	Fonte del tweet	9 valori distinti
set	String	Indica il set di appartenenza del tweet	Due valori distinti: train e test

Tabella 1 Semantica degli attributi

I dati sono divisi in due set: il primo per l'addestramento dei modelli, l'altro per testarne le performance. Il primo comprende l'80% del dataset (3977 record) mentre il secondo è composto dal restante 20% (872 istanze). Per lo svolgimento del sub-task A, la variabile target su cui ci si è concentrati è *irony*, la cui distribuzione, che risulta bilanciata tra le due classi, è osservabile nella Tabella 2.

	Training set
Ironico (1)	2023
Non ironico (0)	1954
Totale	3977

Tabella 2 Distribuzione dei valori nella classe *irony*

[1] https://www.evalita.it
[2] http://www.di.unito.it/~tutreeb/ironita-evalita18/index.html

3. Classificazione

La fase di classificazione si articola in quattro analisi, presentate di seguito. Allo scopo di valutare le prestazioni dei classificatori sono state adottate le seguenti metriche: *Precision*, *Recall*, *F1-score* e *Accuracy*; per ciascuno dei modelli sono, inoltre, state calcolate le matrici di confusione, riportate in ogni sezione specifica. Dato il bilanciamento della variabile target, è stato impiegato un *Dummy Classifier* come *baseline* che, predicendo sempre la classe più frequente, raggiunge il 50.87% di *Accuracy*. Nelle successive sezioni sono presentate le analisi svolte con modelli più sofisticati: le prime tre analisi riguardano l'uso di *LinearSVC* con il parametro *dual* impostato su *False*, che indica l'uso di un algoritmo di ottimizzazione specifico per problemi di classificazione binaria; infine, la quarta analisi prevede l'implementazione di un *Neural Language Model*.

3.1 Prima analisi

La prima parte dell'analisi prevede l'uso di un *Support Vector Classifier Lineare* (*LinearSVC*) che prende in input una rappresentazione del testo basata solo su informazioni linguistiche non lessicali estratte utilizzando il sistema *Profiling-UD*[3].
Prima di procedere alla fase di addestramento, i dati sono stati normalizzati utilizzando un *MinMaxScale*r in modo che tutte le caratteristiche avessero valori compresi tra 0 e 1. Questo passaggio è essenziale per garantire che le features abbiano lo stesso peso nella classificazione.

Per valutare il modello e testarne le capacità di generalizzazione, è stato eseguito un processo di *k-fold cross-validation* (con *k=5)*, che prevede la divisione dei dati di training in cinque fold; per ciascuno dei fold ottenuti è stata calcolata l'accuratezza del modello, riportata nella Tabella 3.

	Accuracy
Fold 1	0.648
Fold 2	0.646
Fold 3	0.666
Fold 4	0.647
Fold 5	0.660

Tabella 3 Risultati della 5-fold cross validation sui dati di training

Le prestazioni del modello sono state valutate attraverso le metriche precedentemente elencate e i risultati, sia sui dati di addestramento che su quelli di test, sono riportati nella Tabella 4 e nella Figura 1.

[3] Strumento per il profiling linguistico dei testi
http://www.italianlp.it/demo/profiling-ud/

	Training set				*Test set*			
	Precision	*Recall*	*F1-score*	*Accuracy*	*Precision*	*Recall*	*F1-score*	*Accuracy*
0	0.66	0.60	0.63	0.65	0.69	0.61	0.64	0.67
1	0.65	0.71	0.68		0.65	0.72	0.68	

Tabella 4 Metriche di valutazione su training e test set, relative alla prima analisi

Come è osservabile, il modello ha ottenuto sui dati di test un'accuratezza del 67% che risulta essere leggermente più alta rispetto a quella ottenuta per i dati di training (65%). Il risultato ottenuto, anche se non altissimo, risulta comunque essere soddisfacente, soprattutto rispetto a quello ottenuto dal *Dummy Classifier*.
Analizzando le altre metriche è possibile, inoltre, notare che il modello ha una buona *Precision* per entrambe le classi; mentre, il valore di *Recall* più alto per la classe 1 suggerisce che il modello ha una maggiore capacità di individuare l'ironia rispetto alla non-ironia.

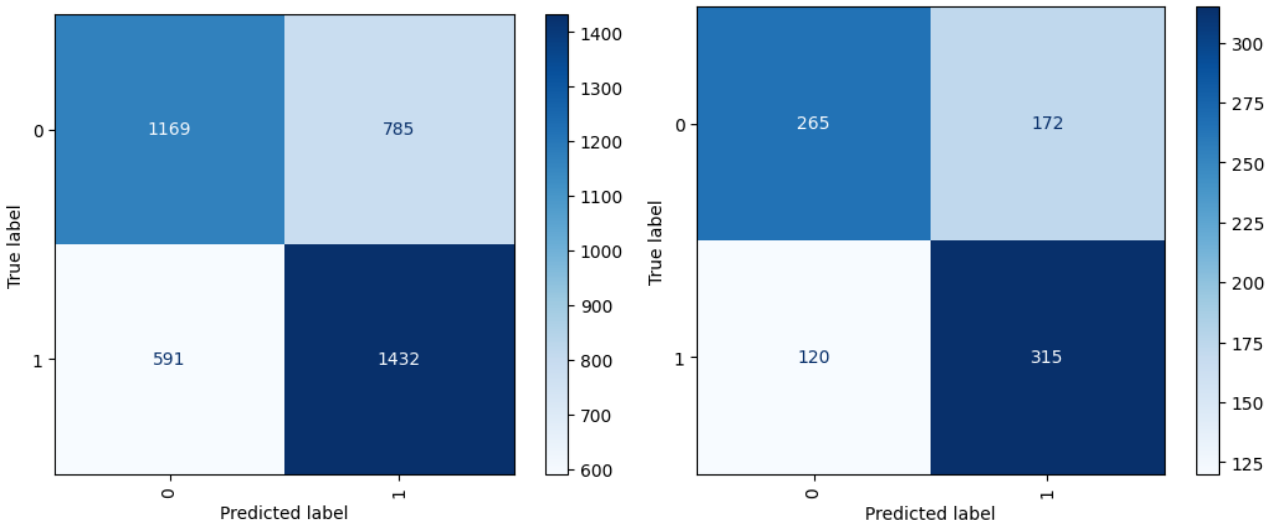

Figura 1 Matrici di confusione su training e test set relative alla prima analisi

In ultimo, è stata calcolata l'importanza delle caratteristiche utilizzate per la classificazione; nello specifico, è stato estratto il vettore dei coefficienti dal modello *SVC*, che rappresenta l'importanza delle features per la classe 0.

In Figura 2 è mostrata l'importanza relativa delle prime 15 features linguistiche e sintattiche per questa analisi.

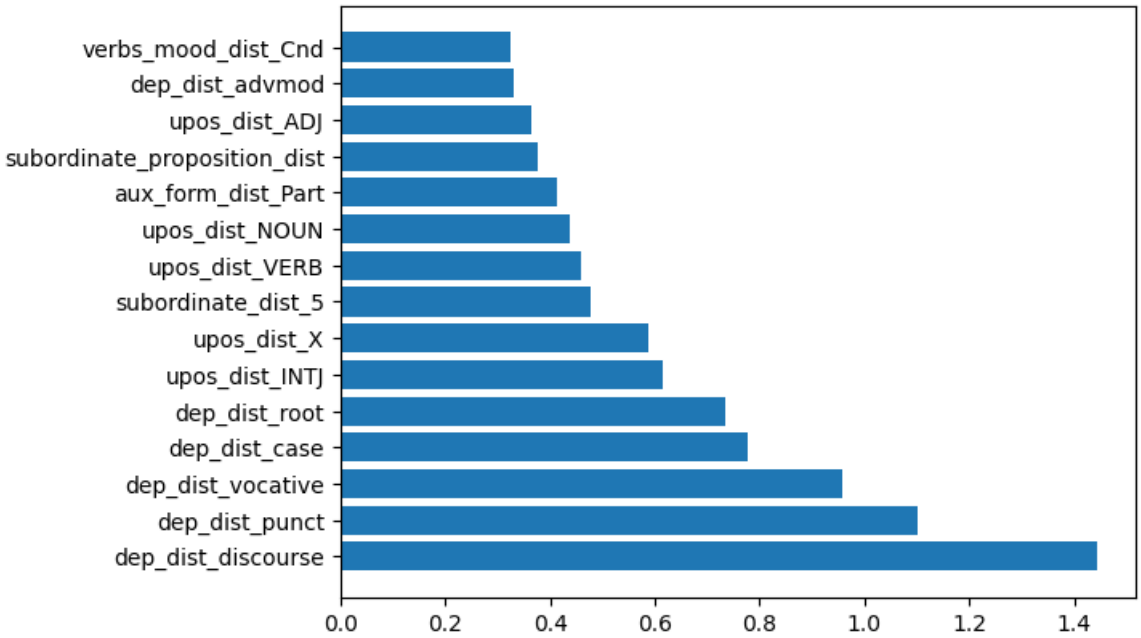

Figura 2 Lista delle 15 features più importanti per la classificazione

È emerso come le caratteristiche che risultano avere la maggiore importanza sono *dep_dist_discourse*, *dep_dist_punct* e *dep_dist_vocative*, le quali fanno riferimento alla distribuzione, rispettivamente, delle relazioni di dipendenza sintattica tra le parole all'interno di un testo o di un discorso più ampio, e di relazioni che coinvolgono segni di punteggiatura e l'uso di espressioni di vocativo all'interno di un testo; seguono poi features legate all'aspetto morfo-sintattico e di subordinazione.

3.2 Seconda analisi

Lo stesso classificatore è stato poi implementato con una rappresentazione del testo basata su n-grammi di caratteri, parole e *Part-of-Speech (PoS)*. Le features sono state estratte da ciascun documento considerando tre casi principali di combinazione di n-grammi:

- Caso 1: estrazione di uni-grammi e bi-grammi di parole e caratteri
- Caso 2: estrazione di bi-grammi e tri-grammi di parole, caratteri e *PoS*
- Caso 3: estrazione di uni-grammi e bi-grammi di parole e *PoS*

Dopo l'estrazione sono state filtrate le features poco frequenti per ridurre l'alta dimensionalità che è stata riscontrata. Per questa operazione di filtering è stata impostata, per ciascun n-gramma, una soglia di frequenza nei testi pari a 5.

I risultati hanno mostrato che il numero di features varia significativamente tra i casi, con il Caso 2 che produce il numero più elevato di features. Il numero di caratteristiche ottenute dopo la fase di filtraggio è riportato di seguito, per ciascun caso:

```
Numero features del train set (Caso 1): 5233
Numero features del test set (Caso 1): 1440
Numero features del train set (Caso 2): 10290
Numero features del test set (Caso 2): 4093
Numero features del train set (Caso 3): 3062
Numero features del test set (Caso 3): 649
```

Prima di procedere alle fasi di valutazione del modello, è stata condotta una fase di pre-processing, consistita nel trasformare le features in una rappresentazione vettoriale e nella loro successiva normalizzazione, utilizzando gli strumenti *DictVectorizer* e *MaxAbsScaler*[4].
Per ciascun caso è stata condotta una cross-validation a 5 fold sul training set e successivamente valutato il modello sui dati di test, sia attraverso i *classification report* che attraverso le matrici di confusione[5]. I risultati ottenuti variano tra i casi ma, nel complesso, tutti e tre mostrano risultati simili sia sul training che sul test set.

Il miglior sistema, rispetto ai risultati ottenuti, è risultato essere il Caso 1, le cui valutazioni sono riportate nelle Tabelle 5 e 6. I Casi 2 e 3 hanno, invece, mostrato dei punteggi di *Accuracy* leggermente più bassi in fase di test, rispettivamente 0.63 e 0.64.

	Accuracy
Fold 1	0.641
Fold 2	0.652
Fold 3	0.659
Fold 4	0.661
Fold 5	0.679

Tabella 5 Risultati della 5-fold cross validation sui dati di training (Caso 1)

	Training set				*Test set*			
	Precision	*Recall*	*F1-score*	*Accuracy*	*Precision*	*Recall*	*F1-score*	*Accuracy*
0	0.66	0.64	0.65	0.66	0.68	0.62	0.65	0.66
1	0.66	0.68	0.67		0.65	0.71	0.68	

Tabella 6 Metriche di valutazione su training e test set, relative alla seconda analisi (Caso 1)

[4] MaxAbsScaler è un metodo di pre-elaborazione dei dati che scala ogni caratteristica dividendo per il valore assoluto massimo di quella caratteristica. Questo processo trasforma le caratteristiche nel range [-1,1].
[5] Non riportate nella relazione per motivi di spazio; sono reperibili nel notebook.

3.3 Terza analisi

Il terzo task di classificazione tramite l'uso di *LinearSVC* è incentrato su una rappresentazione del testo costruita attraverso l'uso dei word embeddings. Per lo svolgimento di questa analisi sono stati impiegati gli embeddings prodotti su *Twitter* e forniti da *ItaliaNLP Lab*[6]. Dopo aver impostato la dimensione degli embeddings a 32, sono state eseguite una serie di normalizzazioni e trasformazioni sul testo, tra cui la gestione dei numeri, l'identificazione degli URL e la formattazione delle maiuscole e minuscole, al fine di preparare il testo per ulteriori analisi o elaborazioni.

Sono state valutate diverse opzioni per l'estrazione delle features e per calcolare le rappresentazioni medie dei tweet basate su varie combinazioni di parti del discorso e tipi di parole. In particolare, le combinazioni prese in considerazione sono le seguenti:

- calcolo della media di tutti gli embeddings, restituendo un unico vettore medio;
- calcolo della media solo degli embeddings di aggettivi, nomi e verbi (altre parti del discorso vengono ignorate); viene restituito un vettore di embeddings medio per tutte queste parole;
- calcolo della media degli embeddings di aggettivi, nomi e verbi; la differenza con la logica precedente consiste nel fatto che il calcolo viene fatto separatamente per ciascuna parte del discorso e successivamente i risultati vengono concatenati in un unico vettore.

Come per le analisi precedenti è stata eseguita una *5-fold-cross-validation*; le variazioni in termini di *Accuracy* ottenuta sui dati di addestramento sono riportate nella Tabella 7.

Accuracy	1 Combinazione	2 Combinazione	3 Combinazione
Fold 1	0.687	0.631	0.611
Fold 2	0.713	0.693	0.653
Fold 3	0.709	0.644	0.654
Fold 4	0.724	0.659	0.626
Fold 5	0.739	0.680	0.670

Tabella 7 Risultati della 5-fold cross validation sui dati di training

Le features estratte con le tre logiche precedentemente elencate sono state normalizzate attraverso un *MinMaxScaler*. Sono stati calcolati i risultati delle metriche di valutazione e delle

[6] http://www.italianlp.it/resources/italian-word-embeddings/

matrici di confusione per ciascuna logica di estrazione delle features, sia per i dati di addestramento che per quelli di test.

Alla luce dei risultati ottenuti, si è scelto di riportare, in Tabella 8 e in Figura 4, solo le valutazioni della prima logica di combinazione (calcolo della media di tutti gli embeddings), la quale ha raggiunto le performance migliori, con un'accuratezza del 71% sui dati di training e del 67% su quelli di test. La seconda e la terza logica di estrazione hanno ottenuto un livello di accuratezza sul test set pari, rispettivamente, al 60% e al 62%.

	Training set				***Test set***			
	Precision	*Recall*	*F1-score*	*Accuracy*	*Precision*	*Recall*	*F1-score*	*Accuracy*
0	0.73	0.67	0.70	0.71	0.71	0.57	0.63	0.67
1	0.70	0.76	0.73		0.64	0.77	0.70	

Tabella 8 Metriche di valutazione su training e test set, relative alla terza analisi

In generale, il modello sembra avere una capacità migliore di individuare tweet ironici (classe 1) rispetto ai tweet non ironici (classe 0), ma a scapito di una *Precision* leggermente inferiore. Migliorare la precisione per i tweet non ironici potrebbe essere un obiettivo per bilanciare meglio le prestazioni del modello. La difficoltà del modello a individuare i tweet non ironici è visibile sia dal basso valore di *Recall* ottenuto (0.57) che dalla matrice di confusione a destra nella Figura 3.

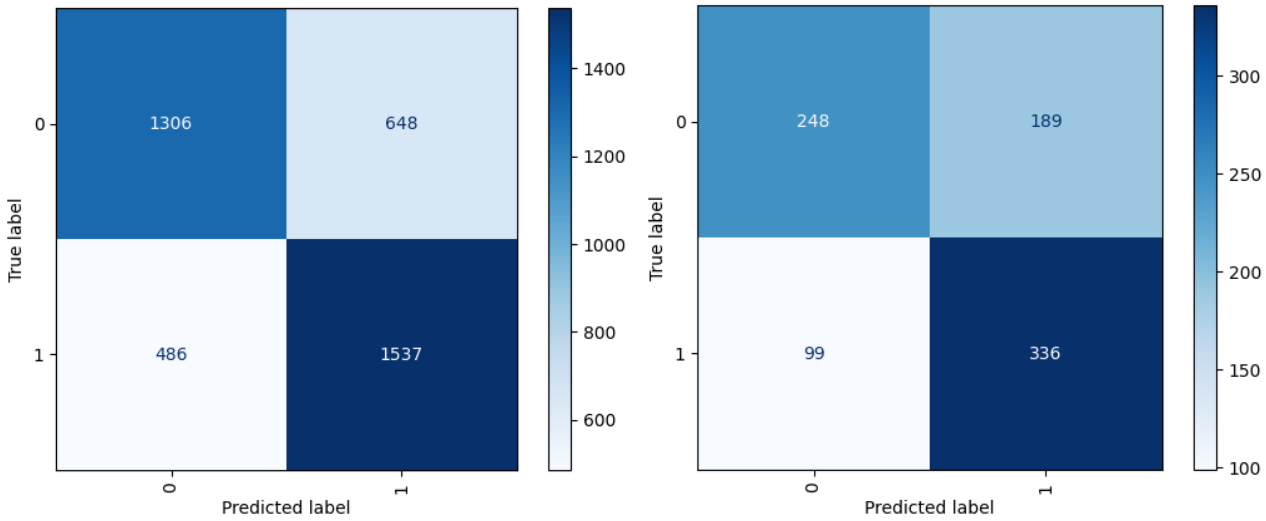

Figura 3 Matrici di confusione su training e test set relative alla terza analisi

3.4 Quarta analisi

Questo studio si concentra sull'addestramento e la valutazione di *Neural Language Model*. Nello specifico sono stati impiegati due modelli di linguaggio pre-addestrati, di tipo *BERT* (*Bidirectional Encoder Representations from Transformers*) e specifici per la lingua italiana:

- *bert-tweet-italian-uncased-sentiment*[7]
- bert-base-italian-cased[8]

È stato effettuato un fine-tuning di 5 epoche sul training set, con una *batch size* pari a *8*, *learning_rate=2e-5* e *weight_decay=0.01*. I dati di addestramento sono stati pre-processati e tokenizzati, pronti per il trainingdei modelli. È stato utilizzato l'F1-score pesato come metrica principale per valutare le prestazioni dei modelli; le sue variazioni, insieme alle variazioni di training e validation loss attraverso le 5 epoche sono riportate in Tabella 9 e 10, rispettivamente per entrambi modelli di addestramento valutati per questo task.

Epoch	Training Loss	Validation Loss	F1-score
1	0.580	0.464	0.758
2	0.419	0.464	0.781
3	0.270	0.955	0.778
4	0.171	1.261	0.777
5	0.09	1.308	0.773

Tabella 9 Valutazione del modello specifico per i tweet attraverso le 5 epoche di addestramento

Epoch	Training Loss	Validation Loss	F1-score
1	0.582	0.483	0.748
2	0.418	0.460	0.781
3	0.282	0.680	0.804
4	0.166	0.883	0.803
5	0.100	0.977	0.793

Tabella 10 Valutazione del modello generale per la lingua italiana attraverso le 5 epoche di addestramento

Nelle Figure 4 e 5 è possibile osservare un ulteriore confronto per la valutazione delle perfromance dei modelli testati. Per ciascuno è stato riportato un *lineplot* della loss, grafico che rappresenta la *loss function* di un algoritmo di machine learning durante le fasi di

[7] https://huggingface.co/osiria/bert-tweet-italian-uncased-sentiment
[8] https://huggingface.co/dbmdz/bert-base-italian-cased

addestramento e di validazione. Sono poi state calcolate le matrici di confusione, sia per i dati di training che di test; per ragioni di spazio, sono state riportate solo quelle calcolate sul test set.

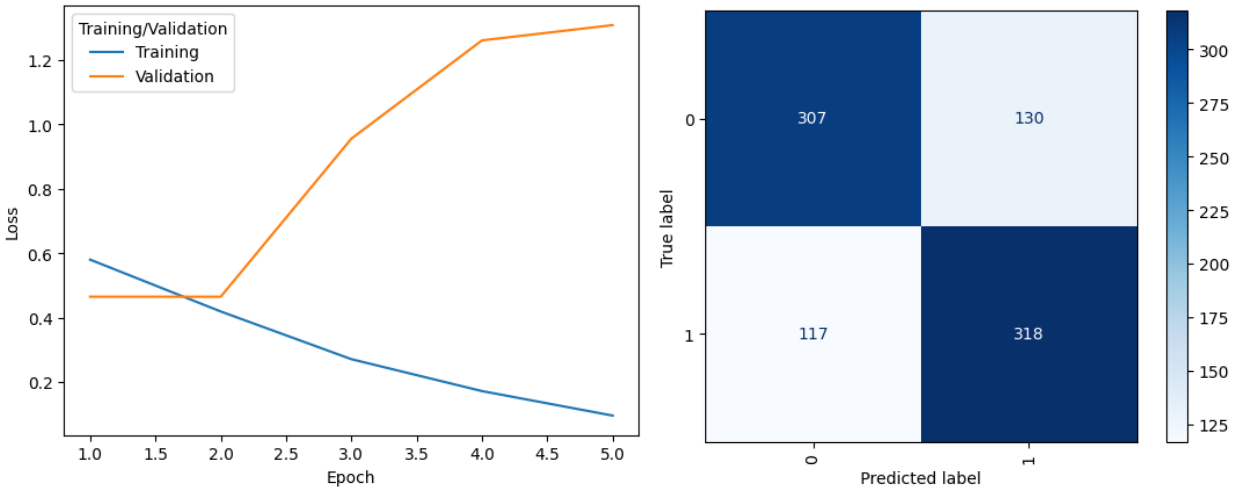

Figura 4 Andamento della *loss function* nelle 5 epoche e matrice di confusione (calcolata su dati di test), relative al modello specifico per i tweet

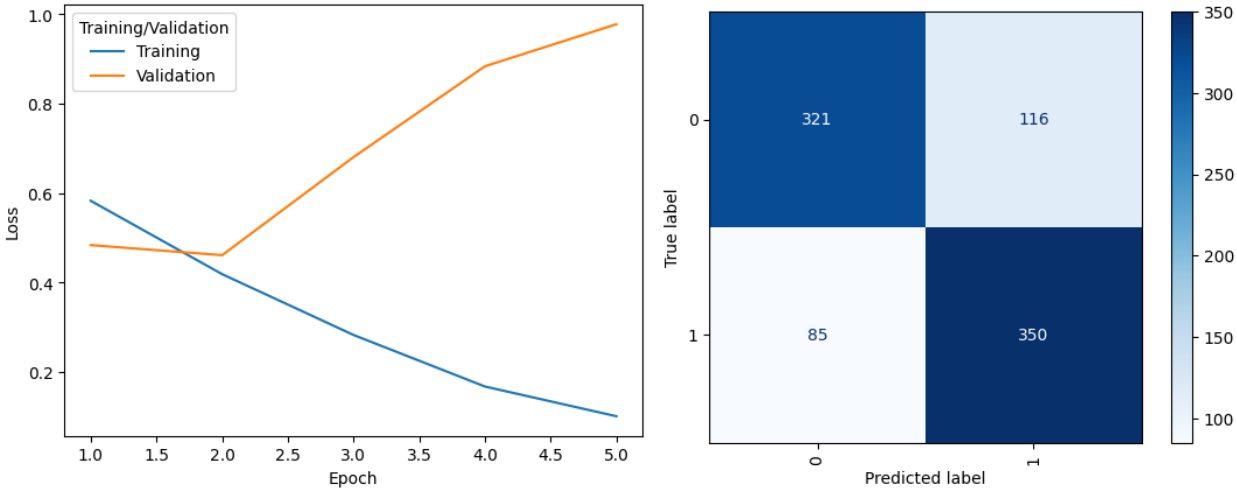

Figura 5 Andamento della *loss function* nelle 5 epoche e matrice di confusione (calcolata su dati di test), relative al modello generale per la lingua italiana

Osservando l'andamento delle curve di loss, è possibile notare come, per entrambi i modelli, la curva di training mostra un andamento discendente. La curva di loss nel caso dei dati di validation risulta, invece, più complicata, in quanto dopo le prima due epoche la curva tende a risalire. Un'interpretazione di un tale comportamento potrebbe essere data dall'overfitting del modello, che tende ad adattarsi troppo ai dati di training e non è capace di generalizzare su nuovi dati. Altre cause potrebbero essere ricercate nello sbilanciamento tra training e validation set, o in problemi di convergenza che richiedono la regolazione dei parametri dei modelli, come il *learning rate*.

Osservando le matrici di confusione precedentemente mostrate, il modello generico per la lingua italiana appare migliore nella predizione corretta di tweet ironici, rispetto al modello addestrato sui tweet (350 predizioni corrette contro 318). Anche rispetto alla predizione della classe opposta risulta essere leggermente più robusto.

	Precision	*Recall*	*F1-score*	*Accuracy*
0	0.72	0.70	0.71	0.72
1	0.71	0.73	0.72	

Tabella 11 Metriche di valutazione sul test set del modello basato sui tweet

	Precision	*Recall*	*F1-score*	*Accuracy*
0	0.79	0.73	0.76	0.77
1	0.75	0.80	0.78	

Tabella 12 Metriche di valutazione sul test set del modello generico sulla lingua italiana

Da un confronto generale, che tiene conto delle metriche di valutazione dei due modelli nel loro complesso, si può constatare come il modello specializzato per i tweet tende ad avere dei risultati più bilanciati tra le predizioni delle due classi. Le performance dei due modelli non differiscono in modo significativo l'uno dall'altro; tuttavia, il modello generico per la lingua italiana ha raggiunto un livello di *Accuracy* più alto (77%) per cui è stato ritenuto più soddisfacente. Nonostante il modello *bert-tweet-italian-uncased-sentiment* sia stato addestrato su testi più specifici e affini al task di interesse è importante tenere in considerazione che l'addestramento è avvenuto su una quantità inferiore di testi rispetto al modello base.

4. Conclusioni

Alla luce delle analisi svolte, il modello che ha raggiunto le performance migliori è stato quello presentato nella Sezione 3.4 che ha interessato l'implementazione di un modello BERT pre-addestrato sul modello *bert-base-italian-cased*, con un'*Accuracy* dello 0.77.

Per la valutazione del task è importante tenere conto della natura stessa dei dati, generati da utenti e raccolti da social media; si parla di testi particolarmente difficili da gestire a causa della loro brevità e dell'analisi fuori dal contesto in cui sono stati generati.
Volendo precedere con sviluppi futuri del lavoro, si potrebbero considerare ulteriori miglioramenti, ad esempio l'ottimizzazione dei parametri dei modelli o l'esplorazione di altri algoritmi di machine learning per vedere se è possibile ottenere prestazioni ancora migliori.

Nel caso dell'analisi svolta tramite *Neural Language Model* si potrebbe tenere in considerazione di utilizzare altri modelli addestrati su quantità di testo più grandi oppure lavorare sul fine-tuning dei modelli implementati in questa analisi, cercando di risolvere eventuali problemi di overfitting tramite l'applicazione di tecniche di regolarizzazione per prevenirlo, come l'aggiunta di dropout o la riduzione della complessità dei modelli.