UNIVERSITÀ DI PISA

MAGISTRALE IN INFORMATICA UMANISTICA

LINGUISTICA COMPUTAZIONALE II

Relazione di progetto:
task IronITA 2018

Simona Sette (544298)

Anno Accademico 2022/2023

Indice

1. Struttura del dataset

Il progetto è stato sviluppato sul task IronITA (Irony Detection in Italian Tweets[1]) di Evalita 2018, con un focus specifico sul subtask A, il cui scopo è identificare se un tweet è caratterizzato da ironia o meno. Il dataset fornito dagli organizzatori si presenta in formato *csv* e contiene le seguenti features:

Features	*Descrizione*
twitter_id	Identificatore del tweet.
text	Contenuto testuale del tweet.
irony	Obiettivo di classificazione: 1 per contenuto con ironia, 0 altrimenti.
sarcasm	Obiettivo di classificazione per subtask B: 1 per contenuto con sarcasmo, 0 altrimenti.
source	Fonte dei tweet.
set	Contiene *training* se il tweet fa parte del set di addestramento, *test* se fa parte del set di test.

Tab 1. Descrizione delle feature del dataset.

Data la distribuzione equilibrata delle due classi (classe *1* con 2023 istanze e classe *0* con 1954 istanze) nel dataset di addestramento e supponendo una strategia di classificazione casuale è possibile affermare che la baseline si attesta intorno al 50%.
L'illustrazione del progetto seguirà la struttura implementata nel codice, la quale si divide in:

- tre parti che identificano le tre macro richieste del progetto;
- il capitolo "Bert" in cui si illustra la realizzazione della richiesta di implementazione di un neural language model.

2. Prima parte

La presente sezione è focalizzata sullo sviluppo di un classificatore support vector lineare da applicare alla rappresentazione del testo basata su informazioni linguistiche non lessicali fornite dal sistema *Profiling-UD*.
Dopo un'iniziale fase di preparazione dei dati, tra cui l'estrazione delle feature e la costruzione di un dataset contenente le feature e le labels per ogni tweet, si è passati a una fase di normalizzazione dei vettori di features tramite *MinMaxScaler*, il quale effettua la normalizzazione scalando i valori in un intervallo tra zero e uno. È poi seguita una fase di splitting del training set in 5 fold al fine di effettuare una 5-cross fold validation e poter sfruttare al meglio le poche istanze disponibili per l'addestramento del modello. Il classificatore implementato è stato *LinearSVC* con parametri di default. Sui singoli fold sono stati ottenuti i valori di accuracy presentati nella tabella in basso a sinistra. Sono stati infine combinati i risultati ottenuti sui singoli fold, che hanno portato ai seguenti risultati definitivi sul training e test set:

Fold	*Accuracy*
1	0.68
2	0.63
3	0.64
4	0.68
5	0.65

	Train Set				**Test Set**			
	Precision	Recall	F1-score	Support	Precision	Recall	F1-score	Support
0	0.66	0.61	0.64	1954	0.69	0.60	0.64	437
1	0.65	0.70	0.67	2023	0.65	0.73	0.69	435
Accuracy			0.66	3977			0.67	872
Macro Avg	0.66	0.66	0.66	3977	0.67	0.67	0.67	872
Weighted Avg	0.66	0.66	0.66	3977	0.67	0.67	0.67	872

Tab 2. Sulla sinistra una tabella che mostra l'accuracy raggiunta sui diversi fold; sulla destra i risultati definitivi della classificazione sul train e test set.

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

Come illustrato nella tabella 2, è stata ottenuta un accuracy dello 0.66 durante il processo di 5-cross fold validation e 0.67 sul test set, il che evidenzia un comportamento generalmente positivo ma non pienamente soddisfacente del modello, come dimostrano le matrici di confusione che seguono: esse evidenziano un comportamento coerente in termini di proporzioni di dati correttamente classificati e no tra set di addestramento e test, con una buona capacità nel riconoscimento delle istanze positive e maggiore incertezza nel riconoscimento della classe negativa.

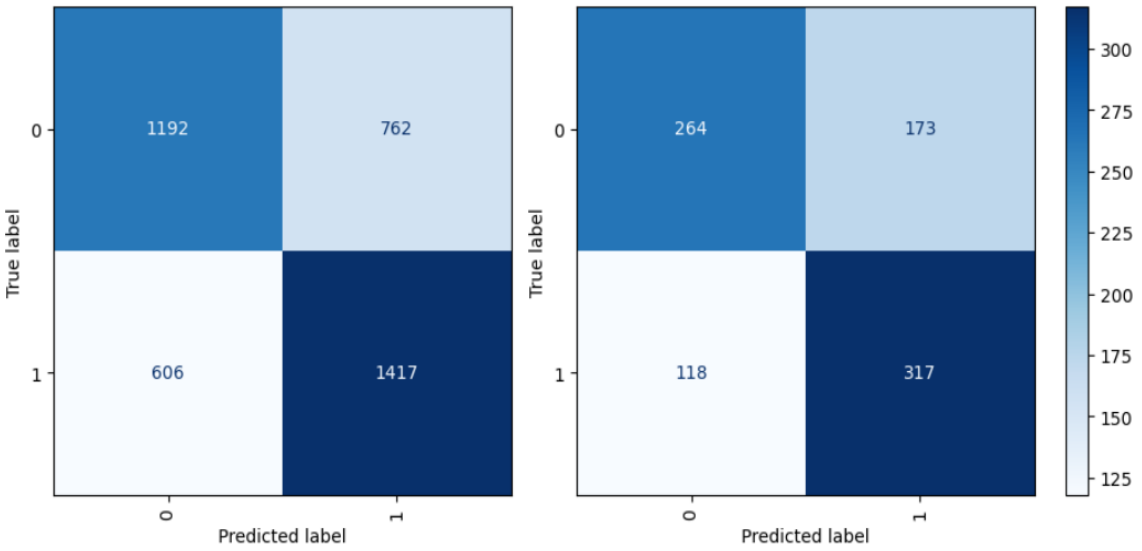

Fig 1. Confusion matrix ottenuta dalla classificazione sul train (sinistra) e test set (destra).

In seguito, tramite l'estrazione dei coefficienti prodotti dal Linear Support Vector Classifier, sono state estratte le 15 feature linguistiche più e meno significative per la discriminazione tra le due classi. Questo processo ha prodotto i seguenti risultati:

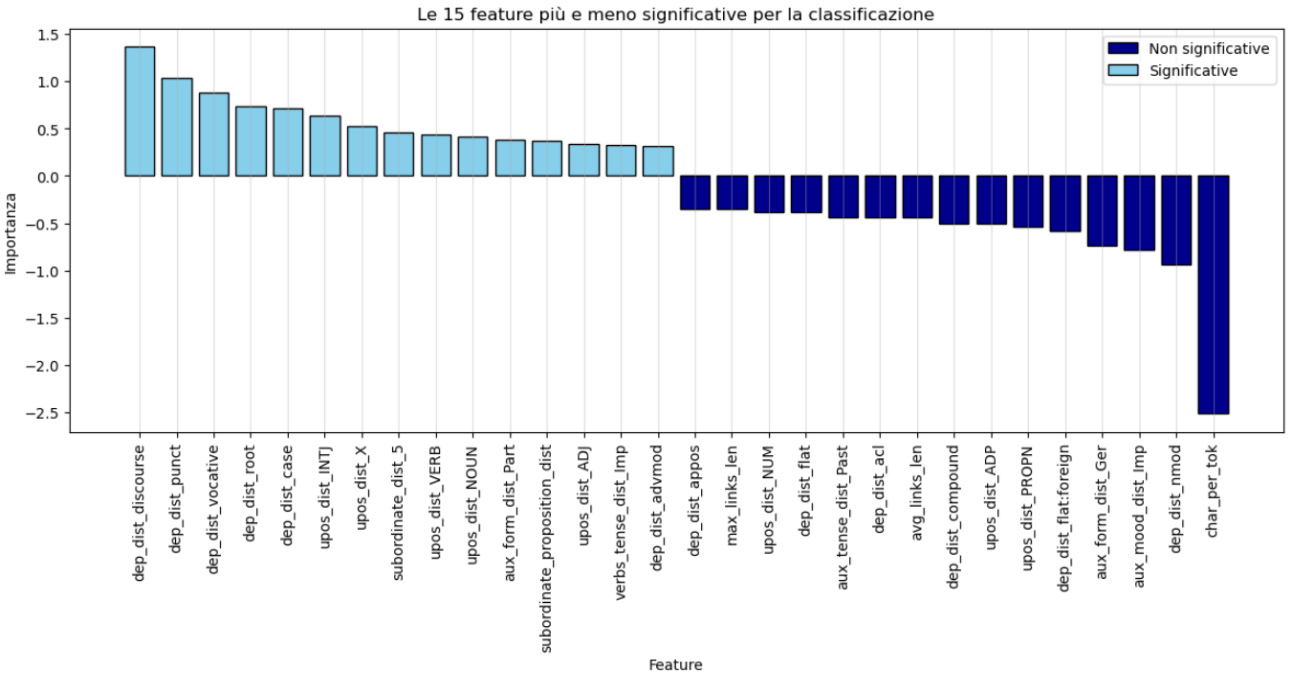

Fig 2. Le 15 feature più o meno discriminanti per la classificazione binaria.

È possibile constatare che tra le prime 15 feature più significative maggiore enfasi viene data alla classe di feature legate alle relazioni sintattiche come *dep_dist_**, le quali si posizionano nelle prime

5 posizioni di importanza, seguite da diverse features legate all'aspetto morfosintattico (upos_dist_*) e di subordinazione (subordinate_*). Al contrario, tra le 15 features meno significative, l'unica caratteristica preponderante in termini di non utilità corrisponde al numero di caratteri contenuti nei token, preceduta da altre feature meno rilevanti ma che si distaccano in modo importante da *char_per_tok* in termini di valore di coefficienti (-2.5 contro valori nel range tra 0 e -1).

3. Seconda parte

L'obiettivo di questa sezione è illustrare lo sviluppo di un classificatore support vector lineare da applicare alla rappresentazione del testo basata su n-grammi di caratteri, parole e part-of-speech. Come effettuato anche per l'esperimento illustrato nella sezione precedente, la fase iniziale di preparazione dei dati ha previsto l'estrazione delle feature tramite l'annotazione linguistica fornita dal sistema Profiling-UD e la costruzione di strutture dati composte da liste contenenti dizionari portatori di informazioni linguistiche (parola, il lemma e la part of speech) per ogni token in un tweet. A questo punto sono state implementate 3 diverse logiche di estrazione delle features al fine di testare come diverse combinazioni e setting di n-grammi possano influire sulla classificazione:

1. Logica di estrazione feature: unigrammi e bigrammi di caratteri e parole;
2. Logica di estrazione feature: bigrammi e trigrammi di caratteri e parole;
3. Logica di estrazione feature: unigrammi e bigrammi di part-of-speech e parole.

A causa dell'alto numero di feature prodotte è stato applicato un filtro di frequenza minima da raggiungere perché vengano considerate significative per la fase di classificazione. Il filtro è stato settato con soglie diverse a seconda della logica di estrazione delle feature al fine di poterne bilanciare il numero definitivo e ottenere una misura standardizzata di partenza per le 3 classificazioni: in particolare,è stato impostato a 3 per la *logica tre*, 10 per la *logica due* e 5 per la *logica uno*. Questi settings hanno prodotto il seguente numero di feature per ciascuna logica:

- Logica 1: 5233 di train, 1440 di test;
- Logica 2: 5236 di train, 2071 di test;
- Logica 3: 6123 di train, 1216 di test.

Dopo aver estratto le feature è seguita una loro fase di trasformazione in una rappresentazione vettoriale tramite *DictVectorizer* e, infine, si sono normalizzati i vettori tramite *MaxAbsScaler*, il quale effettua la normalizzazione basandosi sul valore assoluto massimo. Tramite il metodo *get_feature_names_out* del vettorizzatore è stato infine possibile conservare le feature originali al fine di realizzare un'estrazione di feature importance finale, come fatto per la parte 1. È poi seguita una fase di splitting del training set in 5 fold per effettuare una 5-cross fold validation. Il classificatore implementato è stato *LinearSVC* con unico parametro *dual=False* diverso dal default; è stata fatta questa scelta in quanto è preferibile impostare *dual* su *false* quando il numero di feature è superiore al numero di istanze disponibili. Sui singoli fold sono stati ottenuti i valori di accuracy presentati nella tabella a sinistra.

Fold	*Accuracy Logica 1*	*Accuracy Logica 2*	*Accuracy Logica 3*
1	0.65	0.64	0.67
2	0.65	0.65	0.66
3	0.68	0.66	0.65
4	0.67	0.66	0.67
5	0.63	0.65	0.64

Tab 3. Accuracy raggiunta sui diversi fold per i diversi approcci.

Sono stati infine combinati i risultati ottenuti sui singoli fold, che hanno portato ai seguenti risultati[2] definitivi sul training e test set per le 3 diverse modalità di estrazione:

	Logica 1		**Logica 2**		**Logica 3**	
	Train Set	Test Set	Train Set	Test Set	Train Set	Test Set
Accuracy	0.66	0.66	0.65	0.64	0.66	0.63
Precision (media macro e pesata)	0.66	0.66	0.65	0.65	0.66	0.64
Recall (media macro e pesata)	0.66	0.66	0.65	0.64	0.66	0.63
F1-score (media macro e pesata)	0.66	0.66	0.65	0.64	0.66	0.63

Tab 4. Risultati definitivi della classificazione sul training e test set in base ai diversi modi di estrarre le features.

Come riscontrabile dai risultati, nessun esperimento di classificazione presenta valori particolarmente significativi: il migliore è risultato essere il modello sviluppato sulle feature filtrate composte di unigrammi e bigrammi di caratteri e parole, che ha raggiunto un accuracy dello 0.66 sul test set coerentemente con l'accuracy raggiunta sul train. Al contrario, il modello sviluppato sulle feature, ove applicata la logica 3 di estrazione, ha raggiunto 0.63 sul test nonostante partisse dallo stesso livello di accuracy del modello migliore, risultando quindi il peggiore dei modelli; nonostante ciò si è comunque rivelato il modello le cui feature discriminanti estratte (tramite lo stesso processo affrontato nella prima parte) sono risultate più interessanti da analizzare:

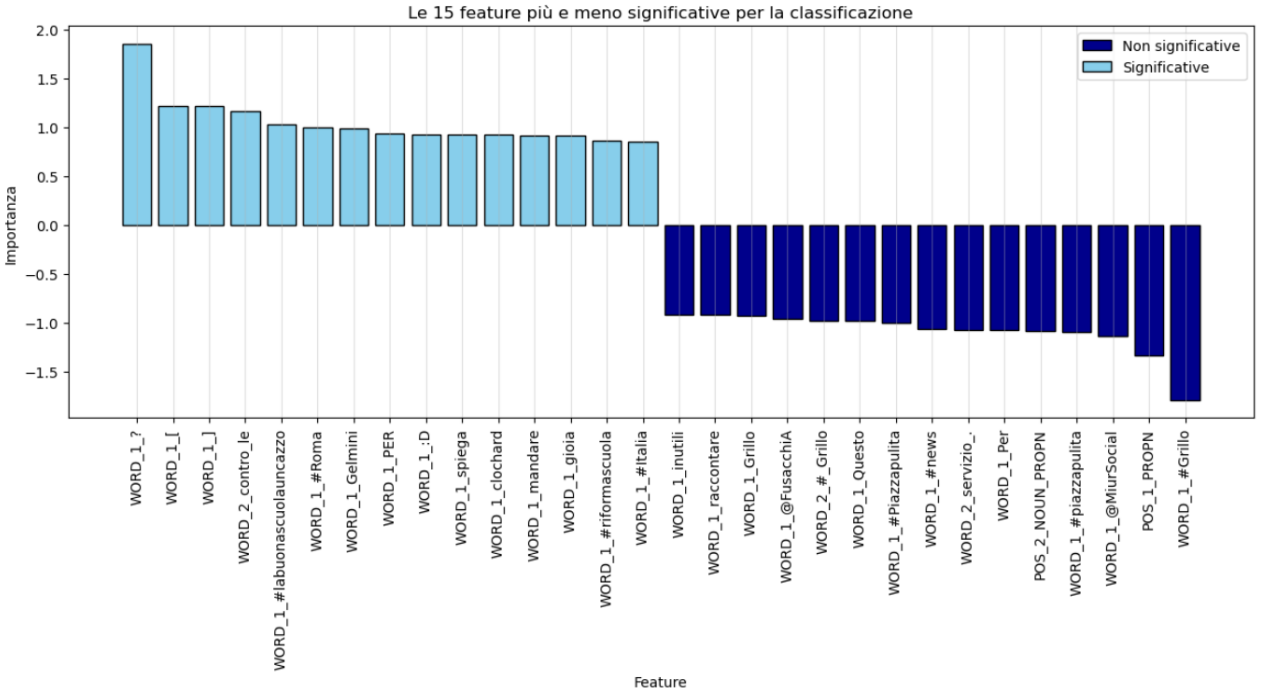

Fig 3. Le 15 feature più o meno discriminanti per la classificazione binaria svolta utilizzando l'approccio di estrazione feature descritto nella logica 3.

Come evincibile dalla *figura 3*, è possibile constatare la presenza di hashtag sia tra le feature discriminanti sia per quelle non discriminanti: tra le feature discriminanti sono presenti unigrammi di parole che rappresentano hashtag ironici/critici che accompagnano contenuti legati alla *riforma*

[2] Per motivi di spazio sono state omesse le confusion matrix ed è stata presentata unicamente la feature importance più interessante: il resto è interamente disponibile all'interno del notebook.

buona scuola[3], come ad esempio "#labuonascuolauncazzo". Sono inoltre presenti sostantivi come "clochard" e unigrammi di parole che rappresentano emoji (es. ":D") e simboli di apertura e chiusura di parentesi quadre, che sono contenuti in molti dei tweet e sono volte a rappresentare le menzioni/tag di altri utenti. Per quanto riguarda le feature non discriminanti, è osservabile come le feature siano strettamente legate all'ambito politico, in quanto compaiono hashtag contenenti nomi di politici e di programmi tv legati alla sfera politica, come *Piazzapulita*: questo potrebbe suggerire il fatto che i temi legati alla sfera politica siano meno importanti per la discriminazione in quanto presenti sia in contenuti ironici che non ironici.

4. Terza parte

Questa sezione è focalizzata sullo sviluppo di un classificatore support vector lineare da applicare alla rappresentazione del testo costruita attraverso word embedding.
Il set di word embeddings adottato[4] è stato *Italian Twitter embeddings*, addestrato specificatamente su tweet in lingua italiana e la dimensione selezionata è stata di 32 per limiti computazionali.
Dopo la preparazione dei testi contenuti nei tweet, al fine di rendere la forma dei token coerente con i token contenuti nel set di word embedding, sono state anche qui definite 3 diverse modalità di estrazione e combinazione degli embeddings:

1. Logica di estrazione feature: calcola la media di tutti gli embedding dei tweet;
2. Logica di estrazione feature: calcola la media degli embedding filtrati per POS, senza fare distinzione fra essi;
3. Logica di estrazione feature: calcola la media degli embedding filtrati in base al POS, facendo distinzione tra aggettivi, sostantivi e verbi.

Dopo aver estratto le feature per ogni approccio si è passati a una fase di normalizzazione dei vettori di features tramite *MinMaxScaler*. È poi seguita una fase di splitting del training set in 5 fold al fine di effettuare una 5-cross fold validation. Il classificatore implementato è stato *LinearSVC* con parametri di default fatta eccezione per il parametro dual, al fine di permettere una convergenza più rapida.
Sui singoli fold sono stati ottenuti i valori di accuracy presentati nella tabella a sinistra.

Fold	*Accuracy Logica 1*	*Accuracy Logica 2*	*Accuracy Logica 3*
1	0.72	0.66	0.65
2	0.71	0.66	0.63
3	0.72	0.67	0.65
4	0.70	0.65	0.63
5	0.72	0.64	0.64

Tab 5. Accuracy raggiunta sui diversi fold per i diversi approcci.

Sono stati infine combinati i risultati ottenuti sui singoli fold, che hanno portato ai seguenti risultati definitivi sul training e test set:

[3] Il che ci suggerisce come lo scraping comprenda tweet probabilmente risalenti a periodi vicini al 2015;
[4] http://www.italianlp.it/resources/italian-word-embeddings/

	Logica 1		Logica 2		Logica 3	
	Train Set	Test Set	Train Set	Test Set	Train Set	Test Set
Accuracy	0.71	0.67	0.66	0.60	0.64	0.61
Precision (media macro e pesata)	0.72	0.68	0.66	0.60	0.64	0.61
Recall (media macro e pesata)	0.71	0.67	0.66	0.60	0.64	0.61
F1-score (media macro e pesata)	0.71	0.67	0.66	0.60	0.64	0.61

Tab 6. Risultati definitivi della classificazione sul training e test set in base ai diversi modi di estrarre le features.

Dai risultati si evince come le migliori performance vengano raggiunte dal modello addestrato sulle medie di tutti gli embedding senza considerare la part-of-speech *(logica 1),* il quale raggiunge un'accuracy del 0.71 sul train set e 0.67 sul test set. Il modello costruito filtrando le part-of-speech ha ottenuto risultati migliori (0.66) sull'addestramento rispetto al modello costruito filtrando le part-of-speech facendo distinzione tra aggettivi, sostantivi e verbi (0.64), ma ha mostrato un comportamento leggermente peggiore dell'altro sul test set (0.60 contro 0.61); infine, entrambi questi modelli hanno dimostrato performance chiaramente peggiori rispetto al primo.

Dalla matrice di confusione costruita sul modello migliore possiamo constatare un ottimo comportamento nella previsione delle istanze appartenenti alla classe positiva (tweet contenenti ironia) e maggiori incertezze nella previsione della classe negativa, dove le prestazioni

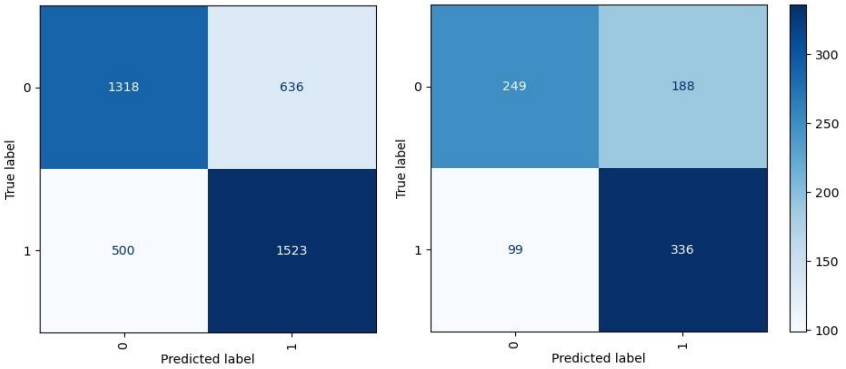

Fig 4. Confusion matrix ottenuta dalla classificazione sul train (sinistra) e test set (destra) utilizzando la logica 1 di estrazione di features.

si dimostrano sufficienti (249 correttamente identificate conto 188 mal classificate) ma non paragonabili a quelle ottenute per classe positiva (336 correttamente identificate conto 99 mal classificate). Il comportamento di questo modello si rivela molto simile a quello realizzato per la *parte 1*, con un sensibile miglioramento nel riconoscimento delle istanze positive e leggero peggioramento per quanto riguarda quelle negative.

5. Bert

In questa sezione finale è stata portata avanti l'implementazione di un neural language model, nello specifico modelli di classificazione basati su BERT (Bidirectional Encoder Representations from Transformers) con un numero di epoche equivalente a 5.
Sono stati implementati due diversi modelli pre-addestrati al fine di poter effettuare una valutazione tra i risultati ottenuti su un modello più specializzato sul tipo di dato testuale (tweets) e un modello addestrato su un vasto numero di dati testuali di forma più generica della lingua italiana.
I modelli implementati sono disponibili su hugging face e sono:

- *bert-tweet-base-italian-uncased*[5]*:* modello addestrato su tweets italiani;
- *bert-base-italian-xxl-uncased*[6]*:* modello addestrato su testi italiani generici;

[5] https://huggingface.co/osiria/bert-tweet-base-italian-uncased
[6] https://huggingface.co/dbmdz/bert-base-italian-xxl-uncased

Dopo una frase di preparazione dei dati testuali, tra cui la definizione dei dataset di train, validation e test e la tokenizzazione delle istanze testuali mediante tokenizzatore di BERT (il quale necessita di una propria pre-elaborazione dei testi), sono state definite le metriche su cui valutare il modello, gli argomenti da passare al trainer e i trainer di entrambi i modelli. Per quanto riguarda la definizione degli argomenti, dopo un'iniziale fase di test di tuning manuali effettuati unicamente sul modello specializzato sui tweets, è stato deciso di fornire gli stessi valori di argomenti per entrambi i modelli al fine di poter effettuare una valutazione comparata delle prestazioni tra un modello specializzato e un modello più generale della lingua. I motivi che hanno portato ad effettuare il tuning unicamente sul modello specializzato sono stati i seguenti:

- limiti computazionali;
- è stata presa la decisione di testare il modello non specializzato e addestrato su un numero maggiore di testi solo successivamente, a causa di risultati non pienamente soddisfacenti ottenuti con il modello specializzato.

I distinti valori di argomenti testati sono stati:

Parametri	Scopo	Candidati
learning_rate	Determina la velocità con cui il modello apprende durante l'addestramento. I valori sono illustrati dal più lento al più veloce.	1e-5, 5e-6, 2e-5, 5e-5, 5e-3, 1e-2, 5e-2, 1e-1
batch sizes (per train e validation)	Determina il numero di esempi di addestramento che vengono propagati attraverso la rete neurale in una singola iterazione durante l'addestramento.	8, 16, 32, 64
weight_decay	Forma di regolarizzazione durante l'addestramento di un modello per prevenire l'overfitting. I valori sono illustrati in maniera decrescente.	0.1, 0.01 , 0.001

Tab 7. Valori candidati per i parametri del trainer.

Per quanto riguarda i valori di learning-rate, si è potuto constatare come con valori superiori di 5e-5, a prescindere della variazione degli altri parametri, i modelli diventano incapaci di imparare a distinguere le classi, classificando o tutti i dati come classe positiva o classe negativa nonostante un buon comportamento delle curve di loss: questo potrebbe indicare la necessità di effettuare un apprendimento moderatamente più lento ma preciso per questo genere di task.
Discorso simile vale per le dimensioni di batch, dove al superamento di soglie come il 32 il modello diventa incapace di generalizzare e raggiungere risultati soddisfacenti.
Come linea generale, osservando le curve di loss dei vari approcci tentati, si è potuto osservare un comportamento non ideale: nella maggior parte dei casi la curva di loss del validation è sempre risultata in salita durante le 5 epoche, mentre quella di train effettivamente ha presentato un andamento discendente. Si è inizialmente ipotizzato che questi comportamenti potessero essere dovuti all'incapacità del modello di imparare durante le prime 5 epoche e sono stati effettuati tentativi con numeri maggiori di epoche (10, 15) che purtroppo non hanno portato ad un miglioramento dell'apprendimento.
Per quanto riguarda i test con 5 epoche, coerenti con la richiesta di progetto, i migliori risultati ottenuti presentano un andamento discendente della curva di loss del validation solo durante le prime 2 epoche, per poi risalire: considerato che questi casi erano spesso accompagnati da discreti valori di accuracy, si è ipotizzato un comportamento di overfitting successivo alle 2 epoche, potenzialmente risolvibile tramite l'impiego del parametro *early stopping,* che non è stato però implementato in quanto il suo utilizzo avrebbe potuto far interrompere l'esecuzione prima delle 5 epoche.
Sono di seguito illustrate alcune delle combinazioni di valori di parametri tentate con i rispettivi valori di accuracy raggiunta sul test set:

Numero setting	*Parametri*	*Accuracy su test set*
1	learning_rate=5e-3, batch_size=32, weight_decay=0.001	0.50
2	learning_rate=1e-2, batch_size=64, weight_decay=0.001	0.50
3	learning_rate=1e-1, batch_size=32, weight_decay=0.1	0.50
4	learning_rate=5e-2, batch_size=32, weight_decay=0.01	0.50
5	learning_rate=2e-5, batch_size=8, weight_decay=0.01	0.69
6	learning_rate=5e-6, batch_size=8, weight_decay=0.01	0.70
7	learning_rate=1e-5, batch_size=16, weight_decay=0.001	0.70
8	*learning_rate=5e-5, batch_size=16, weight_decay=0.01*	*0.72*
9	learning_rate=5e-6, batch_size=16, weight_decay=0.1	0.72
10	learning_rate=2e-5, batch_size=16, weight_decay=0.001	0.72

Tab 8. Combinazioni di valori candidati per i parametri del trainer con relativi risultati in termini di accuracy sul test set.

Una volta ottenuti questi risultati sul modello addestrato specificatamente sui tweet con i parametri illustrati in precedenza, si è passati all'addestramento del modello addestrato su testi italiani non specializzati al fine di poter paragonare le prestazioni dei due modelli a parità di parametri. Le combinazioni di parametri utilizzate sono state le tre che hanno ottenuto i valori di accuracy più alti (0.72). La composizione che ha portato ai migliori risultati su entrambi i modelli è stata la combinazione numero 8 (learning_rate=5e-5, batch_size=16, weight_decay=0.01), portando a un accuracy dello 0.72 sul set di test per il modello specializzato sui tweet e 0.75 sul set di test per il modello di grandi dimensioni non specializzato.
Sono di seguito riportate le metriche di valutazione, le matrici di confusione e le curve di train-validation loss per entrambi i modelli a fine comparativo:

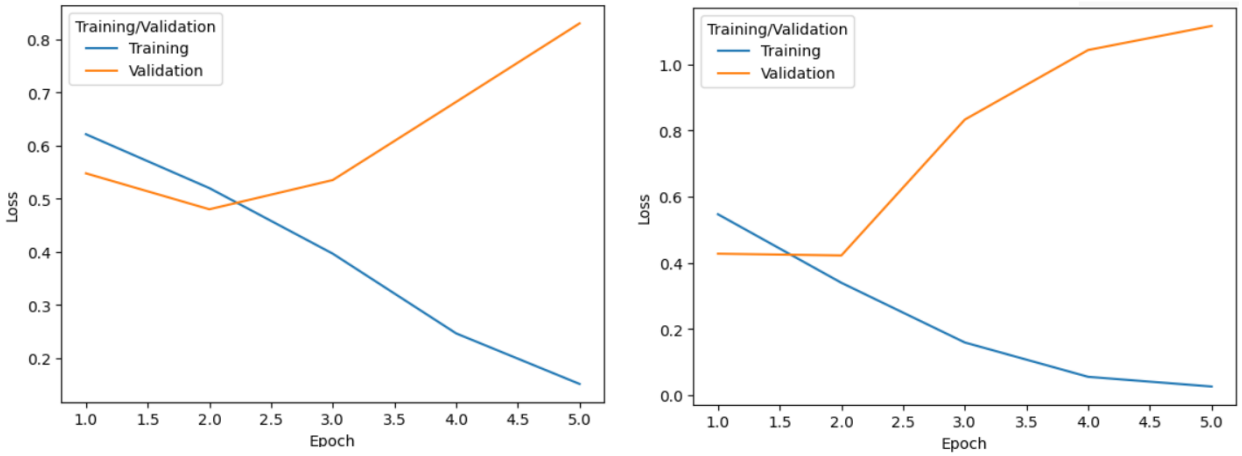

Fig 5. Plot delle curve di loss sul training e validation set per entrambi i modelli (a sinistra quello specializzato sui tweet, a destra quello non specializzato) con la combinazione di parametri numero 8.

Si può osservare in entrambi i plot il comportamento descritto in precedenza per le due curve di loss: il modello riesce ad imparare come raggiungere il suo scopo di classificazione nell'arco di due epoche per poi overfittare ed adattarsi ai dati di training in maniera più specifica. Nonostante la curva di loss sul validation set del modello specializzato mostri una discendenza all'apparenza più marcata rispetto a quella del modello non specializzato, passando da un valore oltre lo 0.5 ad un valore inferiore a questa soglia durante la seconda epoca, in realtà il valore di loss più basso raggiunto è stato quello dalla medesima curva del modello non specializzato, che parte fin dalla prima epoca da un valore che si attesta intorno allo 0.4, per poi scendere quasi impercettibilmente

nella seconda epoca ed in seguito risalire. Nonostante l'overfitting entrambi i modelli mostrano ottime prestazioni sui test set:

	Modello specializzato sui tweet	Modello non specializzato sui tweet
	Test	Test
Accuracy	0.72	0.75
Precision (media macro e pesata)	0.72	0.75
Recall (media macro e pesata)	0.72	0.75
F1-Score (media macro e pesata)	0.72	0.75

Tab 9. Prestazioni sul test set di entrambi i modelli con la combinazione di parametri numero 8.

Come è possibile constatare dai valori delle metriche ottenuti, il miglior modello è risultato essere quello addestrato su un gran numero di dati testuali italiani generici, che ha raggiunto per tutte le metriche il valore di 0.75, rispetto al modello addestrato specificatamente sui tweet che per le medesime metriche ha raggiunto il valore 0.72.
In generale, è possibile constatare quanto all'aumentare dei dati di addestramento le performance dei modelli migliorino, ma in questo specifico caso è anche possibile notare come le performance dei due modelli non differiscono significativamente l'uno dall'altro: questo suggerisce che è possibile ottenere ottimi risultati anche utilizzando modelli addestrati su quantità minori di testo, purché i testi utilizzati appartengano alla stessa tipologia dei testi da classificare.

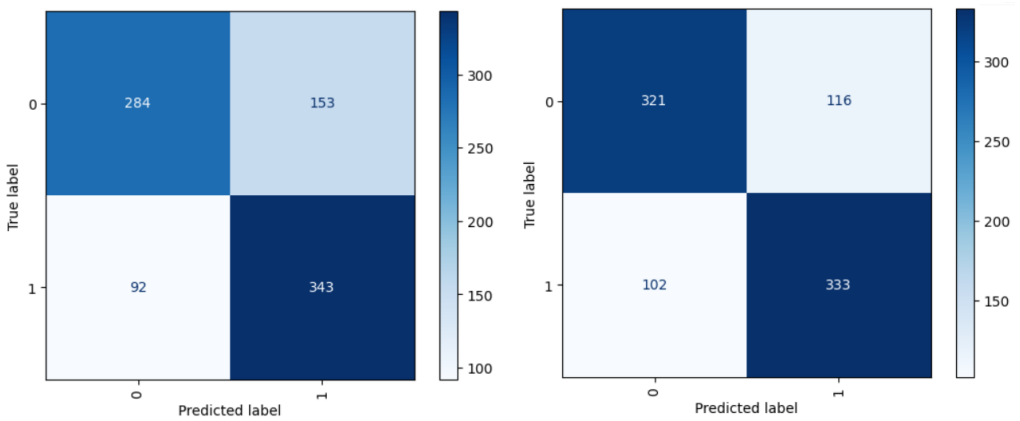

Fig 6. Matrici di confusione realizzate sul test set per entrambi i modelli (a sinistra quello specializzato sui tweet, a destra quello non specializzato).

Infine, le matrici di confusione dei due modelli mostrano come il modello non specializzato sui tweet sia leggermente migliore nella predizioni rispetto al modello specializzato, seppure con le dovute considerazioni: il modello non specializzato pecca nelle predizioni della classe 1 (testo ironico), dove tende a fare predizioni peggiori rispetto al modello specializzato; allo stesso tempo mostra migliori performance nella predizione della classe 0 (testo non ironico), dove tende a fare meno confusione rispetto al modello specializzato.
Per concludere, entrambi i modelli implementati ottengono eccellenti risultati, per quanto il modello non specializzato sui tweet dimostri una leggera superiorità rispetto all'altro.

7. Conclusioni

In conclusione, il modello che ha ottenuto le performance migliori tra tutti quelli testati è stato il BERT pre-trainato su grandi quantità di testo generico in lingua italiana, con un valore di accuracy dello 0.75 sul test set, subito seguito dal BERT pre-trainato su minori quantità di testo ma specializzato sul tipo di testo da classificare, che ha raggiunto lo 0.72 di accuracy sul test set.

Tenendo in considerazione unicamente i migliori modelli identificati dei 3 grandi approcci di classificazione implementati[7] a pari performance (0.67 di accuracy sul test set) abbiamo:

- Il modello Linear Support Vector Classifier addestrato sulle medie di tutti i word embeddings senza considerare la part-of-speech (*Terza parte*);
- Il modello Linear Support Vector Classifier addestrato utilizzando come features le informazioni linguistiche estratte tramite il sistema Profiling-UD (*Prima parte*).

Infine il modello con le performance peggiori, anche se con delle differenze trascurabili rispetto ai due precedentemente esposti (0.66 di accuracy sul test set), è stato il classificatore support vector lineare sviluppato sulle feature composte di unigrammi e bigrammi di caratteri e parole (*Seconda parte*).
In generale, è possibile constatare il miglioramento qualitativo delle performance portato dall'impiego di modelli di classificazione che sfruttano i neural language models, i quali hanno permesso di arrivare a guadagnare quasi fino a 0.1 di valore di accuracy e altre metriche di valutazione rispetto ai modelli Linear Support Vector Classifier implementati utilizzando le diverse tecniche di feature extraction presentate.
Nonostante ciò, la natura stessa del task non ha permesso di raggiungere performance eccellenti in quanto la comprensione dell'ironia è un compito complesso anche per gli umani stessi, dal momento che può variare notevolmente a seconda della lingua, delle norme sociali e del contesto culturale di riferimento, oltre che dalle preferenze e capacità individuali. Gli umani spesso riconoscono l'ironia grazie all' esperienza, conoscenza del mondo e abilità cognitive che permettono di comprendere i doppi sensi, le contraddizioni e le implicazioni sottili all'interno di un testo.
La possibilità di poter fornire tutte le abilità sopracitate ai modelli sta soltanto di recente diventando una potenzialità più vicina alla realtà rispetto al passato, tramite lo sviluppo di neural language models addestrati su un numero colossale di dati testuali: per quanto si possa soltanto parlare in termini speculatori di queste potenzialità future al momento corrente, ciò non toglie che i modelli di apprendimento attualmente disponibili ed adoperabili riescano a raggiungere risultati comunque validi e non trascurabili nella comprensione dell'ironia.

[7] Trattati nel capitolo 1, 2 e 3 (*Prima*, *Seconda* e *Terza parte*) di questa trattazione;