UNIVERSITÀ DI PISA

LINGUISTICA COMPUTAZIONALE 2
HATE SPEECH DETECTION

Irene Mondella (584285)

Anno Accademico 2022/2023

Indice

1 Introduzione

Nel presente lavoro ho sviluppato e valutato diversi modelli di *machine learning* (nello specifico, tre *Suppor Vector Machine* lineari e un *Transformer*) per risolvere il task condiviso HaSpeeDe (*Hate Speech Detection*), organizzato dalla campagna Evalita nel 2018[1]. Si tratta di un task di classificazione binaria: preso in input un post di un social network (in questo caso Facebook), il modello deve riconoscere se esso contiene o meno messaggi di odio.

Per tutti i modelli, l'addestramento è stato eseguito sui 3000 dati etichettati forniti da Evalita: cioè, 3000 post di Facebook etichettati con "1" se contenenti messaggi di odio, con "0" altrimenti.

Nel caso dei *Support Vector Classifier*, i dati di addestramento sono stati suddivisi per 5 volte in *training set* e *validation set* per effettuare un processo di *5-fold Cross Validation* (quindi, di volta in volta, il 20% dei dati viene utilizzato per valutare il modello). In seguito, il test è stato effettuato su un ulteriore dataset formato da 1000 esempi, sempre etichettati.

La *baseline* per questo task su questo dataset è dello 0,539: performance ottenuta da un DummyClassifier che predice sempre la classe più frequente, in questo caso "0".

Il *preprocessing* dei dataset, l'addestramento dei *Support Vector Classifier* e la valutazione dei modelli sono stati effettuati utilizzando la libreria scikit-learn, mentre l'addestramento e la valutazione del *Neural Language Model* è stato effettuato tramite le librerie Evaluate, Datasets e Transformers di Hugging Face.

2 *Support Vector Machine*

Il *Support Vector Classifier* è stato addestrato sui dati di *training* forniti, adottando tre diverse possibili rappresentazioni del testo:

1. Dati testuali rappresentati tramite feature di *profiling* linguistico;
2. Dati testuali rappresentati come occorrenze di n-grammi di parole;
3. Dati testuali rappresentati come *word embeddings*.

Chiaramente, rappresentazioni diverse codificano informazioni diverse, più o meno utili per la risoluzione del task in questione.

2.1 Feature di *linguistic profiling*

Il primo tipo di rappresentazione testato ricorre a 136 feature di *profiling* linguistico: feature che cercano di definire lo stile del testo, anche a livello lessicale, morfosintattico e sintattico. I valori riportati sono stati estratti tramite Profiling-UD[2] e riguardano, ad esempio, la lunghezza media delle frasi, la lunghezza media delle parole, la varietà lessicale, la distribuzione delle categorie grammaticali... Trattandosi esclusivamente di feature numeriche, ognuna di esse è stata normalizzata tramite MinMaxScaler, che riporta ogni valore a un intervallo compreso tra 0 e 1.

Su questo dataset è stato poi addestrato una SVC lineare, limitando la risoluzione del problema di ottimizzazione alla sola forma primale tramite il parametro *dual=False*, da preferire quando il numero di dati è maggiore del numero di feature, come in questo caso. Per quanto riguarda gli altri iper-parametri del modello, solo nel caso di C (iper-parametro di regolarizzazione) è stata effettuata una ricerca del valore migliore, analizzando l'andamento dell'*accuracy* di *training* e *validation* al variare di C (Figura 1).

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

[2] http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.883.pdf

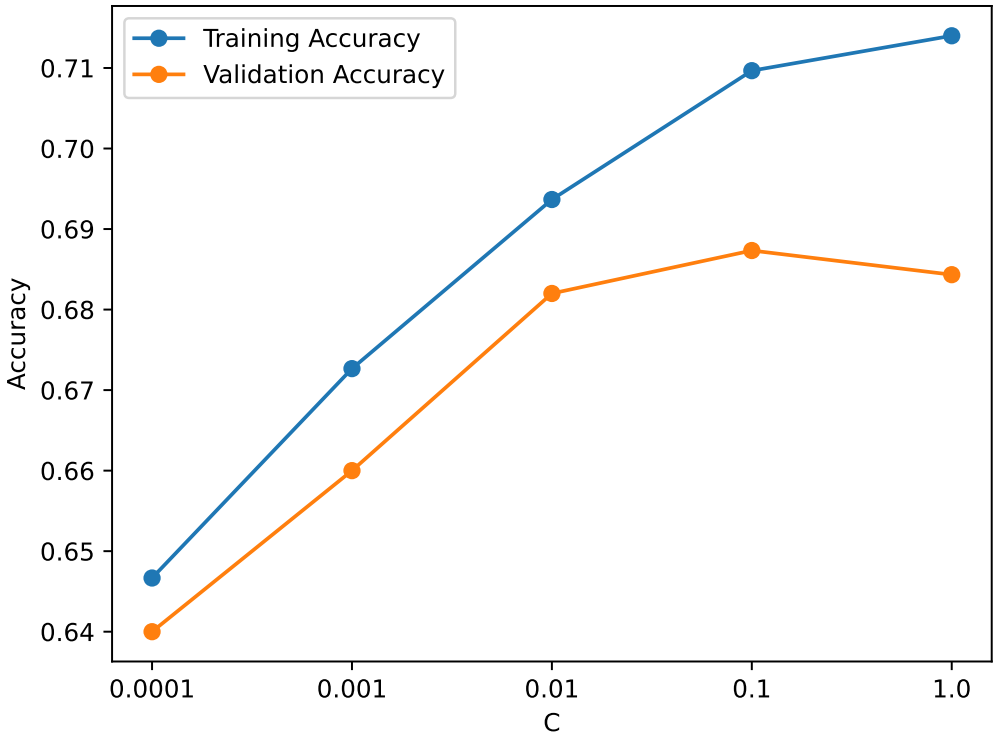

Figura 1: Andamento dell'*accuracy* al variare di C

Come è visibile dal grafico, C=0,1 permette di ottenere l'*accuracy* migliore sul *validation set*, e perciò risulta essere il valore migliore per l'iper-parametro in questione. Nello specifico, la *Support Vector Machine* così impostata ha riportato un'*accuracy* sul *training set* dello 0,71, mentre sul *validation set* ha ottenuto un'*accuracy* media (sulle cinque ripartizioni della *5-fold Cross Validation*) dello 0,687 (con deviazione standard pari a 0,004) e un F1 score medio dello 0,649 (deviazione standard pari a 0,011).

Infine, il modello così addestrato è stato valutato sul *test set*, in cui ha riportato un'*accuracy* dello 0,643 (circa dieci punti percentuali in più della *baseline*) e F1 score di 0,615. Nello specifico, il 61% degli "0" viene classificato correttamente, così come il 66% degli "1", come si può vedere dalla matrice di confusione riportata in Figura 2.

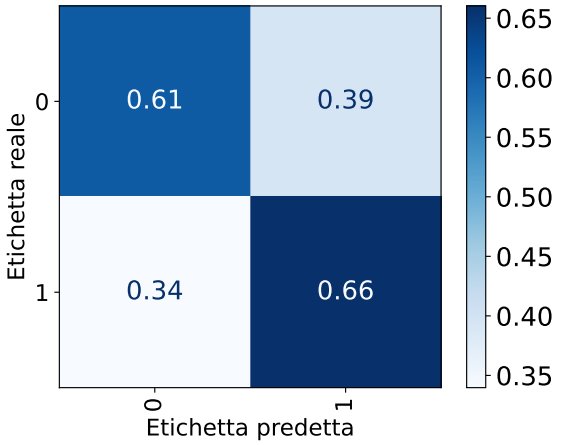

Figura 2: Matrice di confusione del SVC addestrato su feature di *linguistic profiling*

La performance ottenuta non è ottimale, probabilmente a causa del fatto che le feature considerate in questa rappresentazione poco hanno a che fare con la presenza di sentimenti di odio all'interno di un testo.

La Figura 3 mostra che le feature considerate più importanti per la risoluzione di questo task sono: il numero di frasi contenute in ogni post ("n_sentences"), la distribuzione della relazione sintattica UD "*numeric modifier*" ("dep_dist_nummod"), la distribuzione della categoria di *Part of Speech* "*determiner*" ("upost_dist_DET"), la lunghezza media delle sequenze di proposizioni subordinate ("avg_subordinate_chain_len") e altre caratteristiche di questo tipo. Tuttavia, queste feature non discriminano la presenza di un messaggio d'odio, che può comparire anche in testi stilisticamente diversi. Inoltre, il dataset utilizzato non si presenta particolarmente variegato dal punto di vista stilistico, trattandosi esclusivamente di post all'interno di un social network. Questi motivi possono forse spiegare le performance non ottimali ottenute dal modello nel task in questione con questo tipo di rappresentazione.

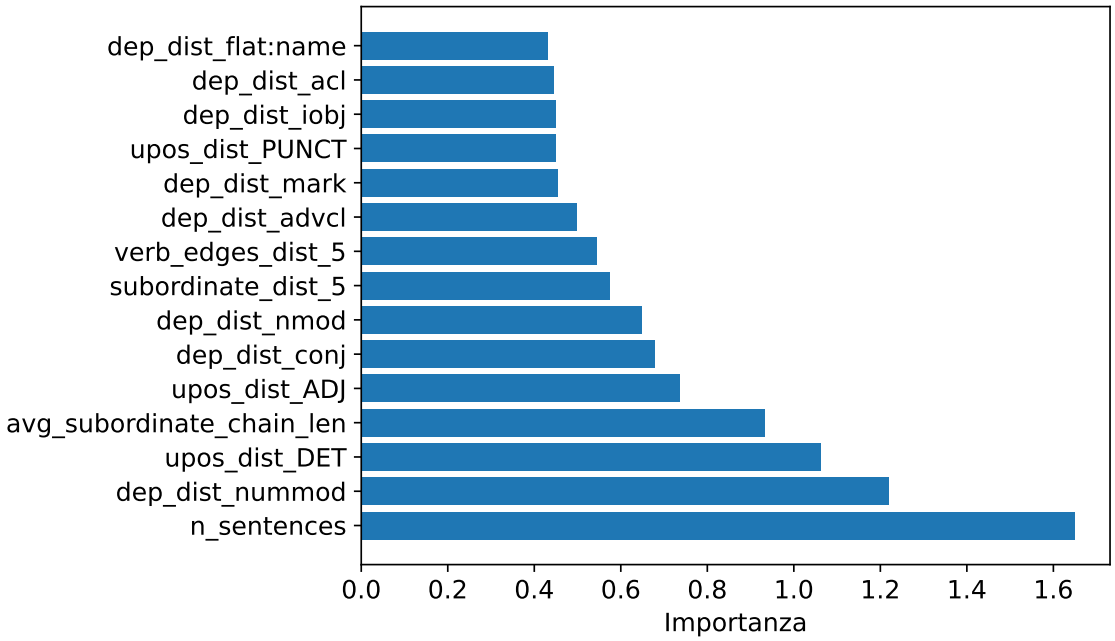

Figura 3: Le 15 feature di *profiling* linguistico considerate più importanti dal SVC

2.2 Feature lessicali

Il secondo tipo di rappresentazione testato ricorre n-grammi di parole, lemmi e caratteri. Nello specifico, sono state estratte le seguenti feature di ogni documento:

- unigrammi, bigrammi e trigrammi di parole;
- unigrammi, bigrammi e trigrammi di lemmi;
- trigrammi di caratteri (in quanto unigrammi e bigrammi di caratteri solitamente non veicolano un significato particolare).

Di esse, sono state mantenute soltanto quelle che ricorrono almeno cinque volte nel *training set*, per un totale di 6085 feature. Il risultato, dunque, è una matrice di dimensioni 3000x6085 estremamente sparsa: ogni documento ha un valore non nullo solo per un numero limitato di feature (quelle che presentano almeno un'occorrenza al suo interno).

Il valore delle feature è stato poi normalizzato: poiché ogni documento presenta una lunghezza diversa, il numero di occorrenze di ogni feature è stato diviso per la lunghezza del documento (in parole o in caratteri in base alla necessità). In seguito, i valori sono stati ulteriormente normalizzati tramite MaxAbsScaler, così da portare ogni valore nell'intervallo [-1, +1].

Sono poi stati valutati 4 diversi *Support Vector Classifier*, per ognuno dei quali è stata effettuata una ricerca del miglior valore di C, analoga a quella descritta nel paragrafo 2.1.

1. Il primo SVC è stato addestrato su tutte le 6085 feature estratte. Il miglior valore di C si è rivelato essere, anche in questo caso, 0,1, con il quale il modello ha ottenuto un'*accuracy* sul *training set* pari allo 0,982, mentre la media delle cinque *accuracy* sul *validation set* dopo la *5-Fold Cross Validation* è dello 0,781 (deviazione standard pari a 0,014), e l'F1 score medio è dello 0,765 (deviazione standard pari a 0,014).

2. Il secondo SVC è stato addestrato solo su unigrammi, bigrammi e trigrammi di lemmi, per un totale di 1591 feature. Essendo il numero di feature inferiore a quello di esempi, è stato impostato dual=False; inoltre, anche qui il valore migliore di C è risultato essere 0,1. Con questi parametri, il modello ha ottenuto un'*accuracy* sul *training set* di 0,861, mentre l'*accuracy* media sul *validation set* è di 0.722 (deviazione standard pari a 0,013), e F1 score medio di 0,687 (deviazione standard pari a 0,012).

3. Il terzo SVC è stato addestrato su unigrammi, bigrammi e trigrammi di parole (1509 feature), con il parametro dual=False e C=0,1. Ha ottenuto un'*accuracy* di *training* pari a 0,862, *accuracy* media sul *validation set* di 0,732 (deviazione standard di 0,008), F1 score medio pari a 0,693 (0,016 di deviazione standard).

4. Infine, l'ultimo SVC è stato addestrato sugli n-grammi di parole e i trigrammi di caratteri (4494 feature). Con dual=True e C=0,1, ha ottenuto un'*accuracy* di *training* pari a 0.963, media di *validation* pari a 0.783 (deviazione standard 0,015) e F1 score medio pari a 0,767 (deviazione standard di 0,018).

Le performance ottenute sono riassunte nella Tabella 1.

Feature utilizzate	*Accuracy training*	Media *accuracy val*	Media F1 *val*
Tutte	0,982	0,781	0,765
Lemmi	0,861	0,722	0,687
Parole	0,862	0,732	0,693
Parole e caratteri	0,963	**0,783**	**0,767**

Tabella 1: *Accuracy* ottenute dagli SVC addestrati su n-grammi

Poiché il quarto modello ha raggiunto la performance migliore sul *validation set*, è stato valutato anche sul *test set*, dove ha raggiunto un'*accuracy* dello 0,72 e F1 score di 0,68. Nello specifico, il 59% degli "0" viene classificato correttamente, così come il 78% degli "1" (Figura 4).

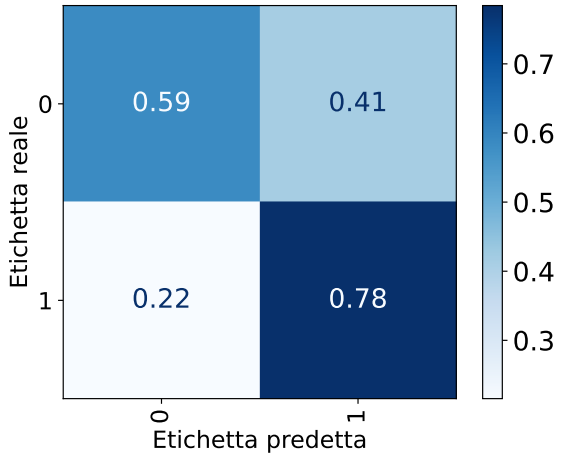

Figura 4: Matrice di confusione del SVC addestrato su n-grammi

Le feature più importanti per la risoluzione del task da parte del modello sono riportante in Figura 5, da cui emerge come sono soprattutto i trigrammi di caratteri a discriminare la presenza di odio o meno all'interno dei documenti, affiancati, in percentuale minore, da due unigrammi di parole, mentre i bigrammi e i trigrammi di parole non sembrano essere considerati rilevanti dal modello per la risoluzione del task.

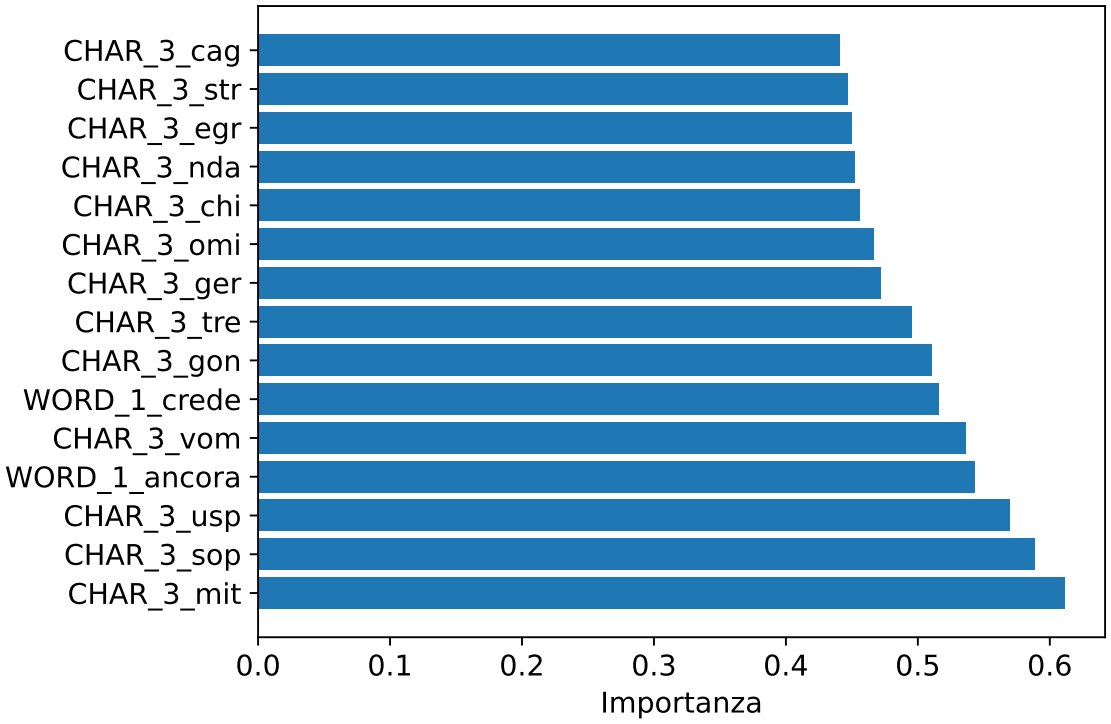

Figura 5: I 15 n-grammi di parole e caratteri considerati più importanti da SVC

2.3 *Word embeddings*

Il terzo tipo di rappresentazione testato ricorre a *embeddings* di parole, di dimensione pari a 128 (itWac *embeddings*[3]). I vettori delle parole di ogni documento sono poi stati aggregati per formare la rappresentazione di quel documento. Sono state usate e valutate diverse tecniche di aggregazione dei vettori; tutte le tecniche considerate, tuttavia, utilizzano soltanto gli *embeddings* delle parole piene, in particolare sostantivi, aggettivi, verbi e avverbi, in quanto le parole funzionali non sembrano particolarmente utili per il task da svolgere.

I tipi di aggregazione di *embeddings* considerati sono i seguenti:

1. media degli *embeddings* di tutte le parole piene (vettore risultante lungo 128);
2. concatenazione degli *embeddings* delle medie, calcolate separatamente, di sostantivi, aggettivi, verbi e avverbi (vettore risultate lungo 512);
3. somma degli *embeddings* di tutte le parole piene (128);
4. concatenazione delle somme, calcolate separatamente, di sostantivi, aggettivi, verbi e avverbi (512);
5. prodotto degli *embeddings* di tutte le parole piene (128);
6. concatenazione dei prodotti, calcolati separamente, di sostantivi, aggettivi, verbi e avverbi (512).

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

I dataset così ottenuti sono poi stati normalizzati tramite MinMaxScaler e utilizzati per addestrare sei diversi SVC lineari. Per ogni modello, è stata effettuata una ricerca del miglior valore di C analoga a quelle condotte in precedenza. Nella Tabella 2 viene riportato, per ogni rappresentazione testata, il valore di C che ha permesso di raggiungere le performance migliori, l'*accuracy* sul *training set* e la media delle *accuracy* sul *validation set*.

	Miglior valore di C	*Training*	Media *validation*
Media	1	0,78	0,751
Concatenazione di medie	0,1	0,825	0,755
Somma	1	0,798	0,774
Concatenazione di somme	0,1	0,822	**0,776**
Prodotto	0,1	0,578	0,566
Concatenazione di prodotti	0,01	0,686	0,643

Tabella 2: *Accuracy* ottenute dagli SVC addestrati su sei dataset diversi

L'*accuracy* migliore, in fase di *validation*, è stata ottenuta dal modello addestrato sulla concatenzazione delle somme degli *embeddings* delle parole piene; perciò, questo SVC è stato provato anche sul test set, anch'esso normalizzato con lo stesso *scaler* usato per trasformare il dataset di addestramento. Qui, il modello ha raggiunto un'*accuracy* dello 0,726 e un F1 score dello 0,71. Nello specifico, il 74% dei post non contenenti odio viene correttamente classificato come 0, e il 72% dei post contenenti odio viene correttamente classificato come 1 (Figura 6).

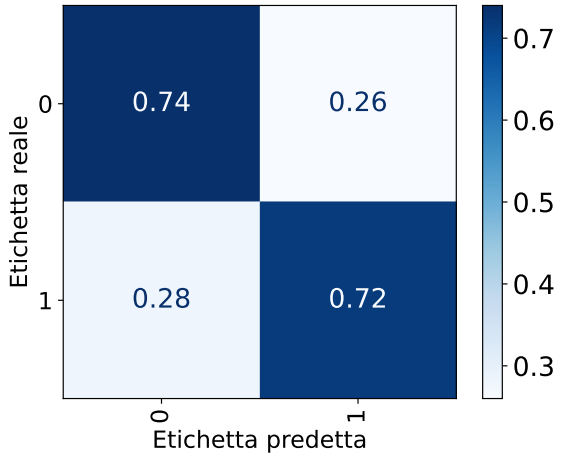

Figura 6: Matrice di confusione del SVC addestrato sulla somma degli *embeddings*

2.4 Risultati dei tre *Support Vector Classifier*

Le performance dei tre SVC testati, addestrati su diverse rappresentazioni del testo, sono riassunte nella Tabella 3.

	SVC-Profiling	SVC-n-grammi	SVC-embeddings
Training	0,71	0,963	0,822
Media *Validation*	0,687	0,783	0,776
Test	0,643	0,72	0,726

Tabella 3: *Accuracy* ottenuta dai tre *Support Vector Classifier* testati

3 Neural Language Model

Infine, è stato testato un *Transformer model*, nello specifico un BERT base preaddestrato sulla lingua italiana e rilasciato dal Munich Digitization Center[4]. Al modello è stato poi effettuato un fine-tuning, addestrandolo sull'80% del *training set*, in quanto i dati restanti sono stati lasciati per la fase di *validation*. Il fine-tuning è stato eseguito per cinque epoche, con una *batch size* pari a 25, un *learning rate* pari a 2e-5 e, come tecnica di regolarizzazione, è stato applicato un *weight decay* con *penalty term* pari a 0,01.

L'andamento delle performance di *training* e *validation* nel corso delle cinque epoche è riportato in Figura 7.

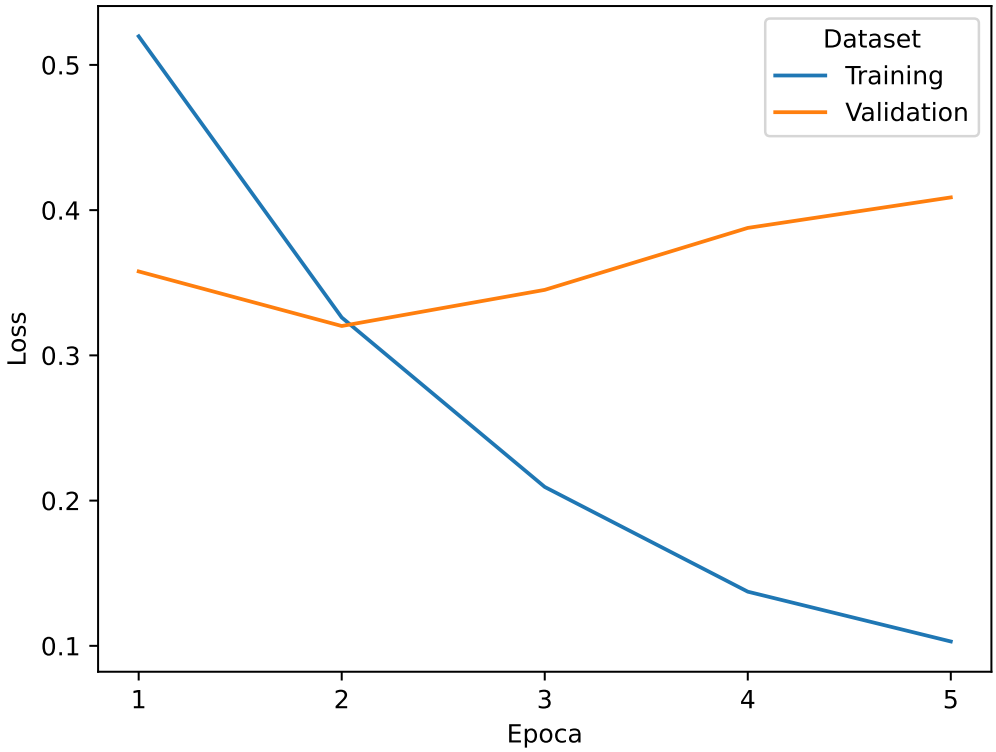

Figura 7: Andamento della funzione di *loss* durante l'addestramento del *Transformer*

Come è possibile constatare dal grafico, a partire dalla seconda epoca il *Transformer* inizia ad andare in *overfitting*: mentre la *loss* sul dataset di addestramento continua a diminuire (segno che il modello si sta adattando sempre di più ai dati), quella sul *validation set* comincia ad aumentare nuovamente. Infatti, come mostra la Tabella 4, la *loss* minore ottenuta in *validation* è pari a 0,327 nella seconda epoca, mentre alla fine dell'addestramento, cioè alla quinta epoca, essa è risalita a 0,409, con un F1 score finale di 0,863.

Epoca	Training Loss	Validation Loss	F1 score
1	0,52	0,358	0,837
2	0,3267	0,327	0,858
3	0,209	0,345	0,866
4	0,137	0,388	0,86
5	0,103	0,409	0,863

Tabella 4: Andamento della performance del *Transformer* durante le 5 epoche di *training*

[4]https://huggingface.co/dbmdz/bert-base-italian-cased

Per evitare l'overfitting, si potrebbero adottare alcune tecniche di regolarizzazione: l'*early stopping*, cioè addestrare il modello per meno epoche (nel caso di questo task due sembrano sufficienti), aumentare il *weigth decay* (in quanto il valore usato, pari a 0,01, non sembra regolarizzare abbastanza) o effettuare l'addestramento con il *Dropout*.

Il modello ottenuto è stato poi testato sul *test set*, dove ha raggiunto un'*accuracy* pari allo 0,84 e un F1 score dello 0,81. Nello specifico, in Figura 8 si vede che il 74% dei post non contenenti odio è stato correttamente classificato come "0", mentre l'88% dei post contenenti odio è stato correttamente classificato come "1".

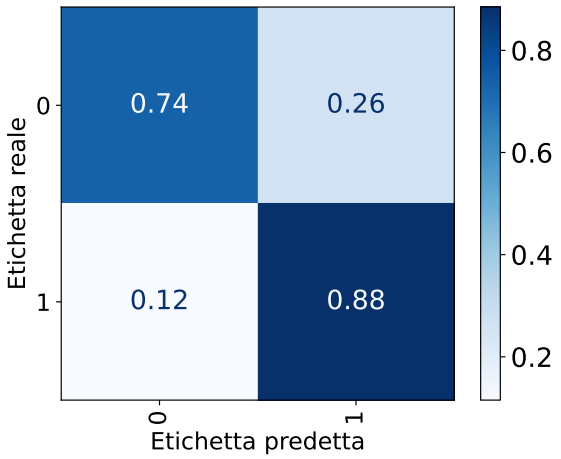

Figura 8: Matrice di confusione del *Transformer model*

4 Conclusioni

Riportiamo in Tabella 5 i valori di *accuracy* ed F1 score dei tre *Support Vector Classifier* e del *Neural Language Model* testati : sia quelli ottenuti in fase di *validation* (che per gli SVC è stata una *5-fold Cross Validation*), sia quelli ottenuti in fase di test.

	Validation		Test	
	Accuracy	F1 score	*Accuracy*	F1 score
SVC-profiling	0,684	0,649	0,647	0,62
SVC-n-grammi	0,766	0,749	0,701	0,665
SVC-embeddings	0,774	0,733	0.73	0,705
Transformer	0,86	0,865	0.83	0,8

Tabella 5: Performance ottenute dai quattro modelli testati

Da questi esperimenti emerge che il *Transformer* raggiunge le performance migliori sia in *validation* che in test. Invece, l'SVC addestrato su *embedding* di parole ottiene risultati di *validation* simili a quelli dell'SVC addestrato su n-grammi, mentre il classificatore addestrato su feature di *profiling* linguistico sembra non generalizzare altrettanto bene.