UNIVERSITÀ DI PISA

Hate Speech Detection
EVALITA20

Mirea Bravaccini

Università di Pisa
Gennaio 2024

1 Introduzione

La presente relazione documenta il lavoro svolto nel contesto del task HaSpeeDe (Hate Speech Detection) proposto nell'ambito di EVALITA 2020 concernente l'individuazione automatica del discorso d'odio. L'obiettivo del lavoro consiste in una classificazione binaria che punta a determinare la presenza o l'assenza di contenuti offensivi nel testo rivolti ad un determinato target (immigrati, musulmani, etc.). Nella relazione sono documentati e discussi i diversi modelli sviluppati, nonché le tecniche di pre-processing e di rappresentazione dei dati e gli algoritmi di classificazione adottati. Per ogni modello si riportano gli esperimenti condotti e le metriche di valutazione ottenute sul test set.

2 Dataset

Il dataset di HaSpeeDe 2020 include testi indirizzati a minoranze come immigrati, musulmani e comunità rom, raccolti tramite tweet e titoli di giornali usando specifiche parole chiave.

2.1 Distribuzione e Formato dei Dataset

Il set di addestramento include 6839 tweets, quello di test 1262, con distribuzione come segue:

Table 1: Distribuzione nel Set di Addestramento

Dataset	HS	Non HS	Tot
Training	2766	4073	6389
Tweet Test	621	641	1262

Entrambi i set sono forniti in formato TSV e contengono, per ogni record, le seguenti features:

1. **ID** del tweet, identificatore univoco numerico;
2. **Testo** del tweet;
3. **Label** di *Hate Speech* (1 se presente, 0 altrimenti);
4. di *Stereotipo*, non utilizzata avendo deciso di sviluppare modelli esclusivamente per il main task (Hate Speech Detection).

3 Modelli

Per questo lavoro sono stati implementati diversi modelli, di cui si riportano nelle sottosezioni successive le metriche concernenti le performance su train e test set. Come baseline è stato utilizzato un classificatore ZeroR con strategia

'most-frequent'[1]. Di seguito si riportano le metriche concernenti la performance del suddetto classificatore, da intendersi come baseline:

	Precision	**Recall**	**F1-Score**	**Support**
0	0.51	1.00	0.67	641
1	0.00	0.00	0.00	621
Accuracy			0.51	1262
Macro Avg	0.25	0.50	0.34	1262
Weighted Avg	0.26	0.51	0.34	1262

Table 2: Risultati della classificazione con ZeroR classifier

In sintesi, sono stati implementati:

1. Tre modelli basati su **SVM lineari**[2]. Per ciascuno, i migliori *hyperparameter* sono stati selezionati tramite un processo di k-fold cross validation (n_folds = 5) e, a seguito dei vari esperimenti, è stato selezionato il modello con *accuracy* superiore.
   (a) Il primo modello è stato addestrato con rappresentazioni del testo basate sulle *feature linguistiche non lessicali* estratte tramite Processing-UD;
   (b) Il secondo è stato addestrato con rappresentazioni del testo basate su *n-grammi di parole e/o caratteri*, selezionato tra diversi modelli differenti tra loro per lunghezza degli n-grammi e tipologia di informazione contenuta;
   (c) Il terzo è stato addestrato con rappresentazioni del testo basato su *embeddings* di parole, differenti tra loro per le parole tenute in considerazione e la tecnica di aggregazione.
2. Un modello basato su **neural language models** usando **bert_uncased_L-12_H-768_A-12_italian_alb3rt0** (meglio noto come alb3rt0).

3.1 Pre-processing

Non si è ritenuto necessario effettuare particolari pre-processing sui testi dei set. L'approccio utilizzato è, complessivamente, conservativo: per tutti i modelli si è deciso di conservare token quali '@user' e 'URL', ritenendo che la loro distribuzione potesse differire tra tweets con label `1` e tweets con label `0`. Questa intuizione è sostanziata, soprattutto per gli URLs, dall'osservazione condotta sul training set, dove è stata rilevata la seguente distribuzione:

[1] Un classificatore ZeroR con strategia 'most frequent' è un semplice modello di machine learning che assegna a tutte le istanze di test la stessa classe, ovvero la classe più frequente nel set di addestramento.

[2] Modello di apprendimento automatico utilizzato per la classificazione di dati binari, dove l'obiettivo è trovare un iperpiano che massimizza la distanza tra i punti di dati delle due classi. Questo iperpiano viene poi utilizzato per prendere decisioni di classificazione su nuovi dati in base alla loro posizione rispetto allo stesso.

Table 3: Distribuzione nel Set di Addestramento

Token	HS	Non HS
'URL'	838	2435
'@user'	1275	1291

Gli hashtag sono invece stati modificati, rimuovendo l'hash ('#') davanti alle parole, sospettando possibili benefici conseguenti questa operazione soprattutto nei modelli basati su embedding (dove questa procedura incrementerebbe il numero di match tra i token dei set e gli embedding disponibili). Ancora, per i modelli dipendenti dagli embedding disponibili su italianlp.it[3] sono state seguite le indicazioni fornite dagli sviluppatori, riassunte di seguito:

1. Numeri: i numeri interi compresi tra 0 e 2100 sono mantenuti nella loro forma originale. Ogni numero intero maggiore di 2100 viene mappato in una stringa che rappresenta il numero di cifre necessarie per memorizzarlo (esempio: 10000 → DIGLEN_5). Ciascuna cifra in una stringa che non è convertibile in un numero deve essere sostituita con il seguente carattere: @Dg. Ecco un esempio di sostituzione (esempio: 10,234 → @Dg@Dg,@Dg@Dg@Dg).

2. Parole: una stringa che inizia con una lettera minuscola deve essere convertita in minuscolo (ad esempio: "aNtoNio" → "antonio", "cane" → "cane"). Una stringa che inizia con una lettera maiuscola deve essere capitalizzata (ad esempio: "CANE" → "Cane", "Antonio" → "Antonio").

3. URLs: i token segnaposto 'URL', presenti nei testi forniti da EVALITA, sono sostituiti da '__URL__' per garantire il match con gli embedding forniti.

3.2 SVM lineari

3.2.1 Modello basato su feature linguistiche non lessicali

Questo modello utilizza rappresentazioni testuali basate sulle feature linguistiche estratte utilizzando Profiling-UD[4]. Questo strumento consente di estrarre più di 130 feature rappresentative della struttura linguistica sottostante del testo dall'output dei diversi livelli di annotazione linguistica (effettuata automaticamente da UDPipe). I fenomeni linguistici modellati possono essere suddivisi in:

1. Proprietà del Testo Grezzo

[3]La risorsa è consultabile all'indirizzo http://www.italianlp.it/resources/italian-word-embeddings/, per ulteriori approfondimenti consultare Cimino et al. (2018)

[4]Profiling-UD è uno strumento e un insieme di risorse sviluppati per il Profiling Uncovered Domains (Profiling-UD) shared task. Per ulteriori informazioni sullo strumento si rimanda a Brunato et al.. Una demo dello strumento è disponibile all'indirizzo http://linguistic-profiling.italianlp.it

2. Variazione Lessicale
3. Informazioni Morfo-sintattiche
4. Struttura del Predicato Verbale
5. Strutture dell'albero di analisi globale e locale
6. Relazioni Sintattiche
7. Uso della Subordinazione

Una volta ottenuto il file in formato CSV (restituito in output dallo strumento), i dati vengono normalizzati utilizzando MinMaxScaler[5]. Il modello utilizza il Grid Search per cercare i migliori iperparametri per il classificatore LinearSVM, in particolare sono testati diversi valori di 'C'[6] e 'dual'[7]. Gli iperparametri individuati ed utilizzati sono:

1. 'C': 3.0
2. 'dual': True

Il modello è poi valutato tramite un processo di 5-fold cross-validation di cui si riporta l'accuracy media e l'accuracy per fold(12). Si riportano nella tabella 5

Fold	Accuracy
1	0.68
2	0.67
3	0.68
4	0.68
5	0.67
Mean Accuracy	0.67
Standard Deviation	0.01

Table 4: Accuracy per fold e accuracy media

anche i risultati ottenuti sul test set. Il modello performa leggermente meglio di uno ZeroR classifier. Possiamo infine osservare quali sono le 15 feature più importanti per la classificazione nell'immagine 1. Interessante notare come le feature più rilevanti siano, nel nostro caso, la distribuzione della relazione flat:name, il numero di frasi, la distribuzione della punteggiatura, il numero di caratteri per token, la densità lessicale.

[5]Tecnica di pre-elaborazione dei dati che normalizza le caratteristiche di un set di dati, ridimensionando i valori in un intervallo tra 0 e 1. Questo processo assicura che le caratteristiche abbiano una scala uniforme, requisito essenziale per il corretto funzionamento di algoritmi sensibili alle differenze di scala come il LinearSVM).

[6]C regola il trade-off tra adattamento ai dati di addestramento e capacità di generalizzazione. Con valori più alti favoriscono adattamento migliore ai dati di addestramento ma possono portare ad overfitting; valori più bassi di C favoriscono la generalizzazione ma possono comportare underfitting.

[7]Il parametro dual determina la formulazione matematica utilizzata nell'addestramento. Quando è True, si utilizza una formulazione duale, adatta per problemi con molte osservazioni e poche variabili. Quando dual è False, si utilizza una formulazione primale, adatta per problemi con molte variabili e poche osservazioni.

Class	Precision	Recall	F1-Score	Support
0	0.60	0.57	0.58	641
1	0.58	0.62	0.60	621
Accuracy			0.59	1262
Macro Avg	0.59	0.59	0.59	1262
Weighted Avg	0.59	0.59	0.59	1262

Table 5: Performance sul test set

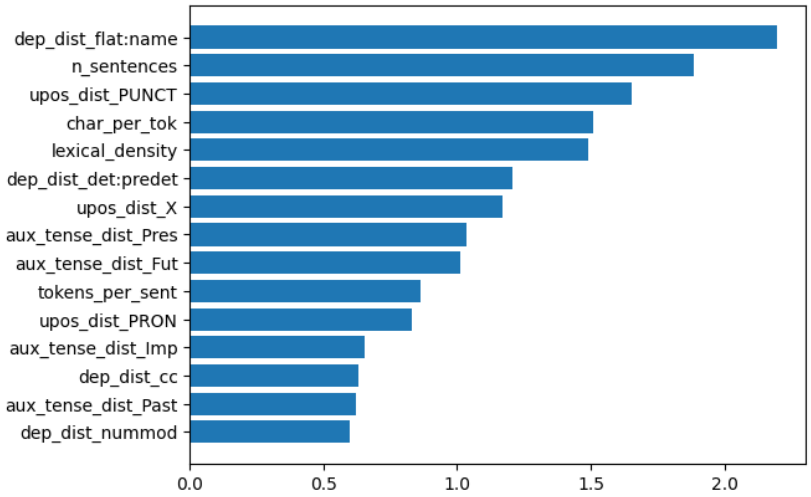

Figure 1: 15 feature più importanti per la classificazione basata su feature linguistiche non lessicali

3.2.2 Modello basato su n-grammi

Questo modello utilizza rappresentazioni testuali basate sul numero di occorrenze di n-grammi di parole[8] e/o caratteri. Per la realizzazione delle rappresentazioni viene utilizzato, come punto di partenza, il file .conllu risultante dall'annotazione effettuata tramite il tool Processing-UD, attraverso il quale possiamo costruire il dataset contenente, per ogni documento, token e relativi lemmi e pos. A questo punto una frase processata appare come segue:

```
['word': 'Diventiamo', 'lemma': 'Diventare', 'pos': 'VERB',
     'word': 'tutti', 'lemma': 'tutto', 'pos': 'DET',
      'word': 'rom', 'lemma': 'rom', 'pos': 'NOUN',
        'word': '(', 'lemma': '(', 'pos': 'PUNCT',
     'word': 'senza', 'lemma': 'senza', 'pos': 'ADP',
    'word': 'nulla', 'lemma': 'nulla', 'pos': 'PRON',
  'word': 'togliere', 'lemma': 'togliere', 'pos': 'VERB',
```

[8]Un n-gramma di parole è una sequenza continua di n parole in un testo

```
 'word': 'a', 'lemma': 'a', 'pos': 'ADP',
 'word': 'i', 'lemma': 'il', 'pos': 'DET',
'word': 'rom', 'lemma': 'rom', 'pos': 'NOUN',
 'word': ')', 'lemma': ')', 'pos': 'PUNCT']]
```

Le feature sono poi estratte, combinando i token (e le informazioni relative) variamente. Gli esperimenti condotti differiscono tra loro per la lunghezza delle sequenze, la selezione delle informazioni da considerare e la combinazione delle precedenti. Le feature, una volta estratte, sono state filtrate per ridurre la dimensione di input (selezionando solo le feature con un numero minimo di occorrenze) e attenuare la natura sparsa della matrice risultante. Questa operazione tuttavia sembra non avere avuto effetti significativi sulle prestazioni del modello e l'esperimento con accuracy media maggiore è stato condotto usando feature non filtrate per numero minimo di occorrenza. La rappresentazione numerica dei testi è poi ottenuta, partendo da questi dizionari di frequenze, con il DictVectorizer e successivamente scalata con MaxAbsScaler. Si riportano di seguito le feature utilizzate per ciascun esperimento e l'accuracy media ottenuta nel processo di 5fold cross validation. Tutti i modelli ottenuti sono realizzati impiegando iperparametri ottenuti tramite GridSearch combinato con kfold cross validation con 5 fold.

	Features	**Acc. Media**
1	Unigrammi, bigrammi e trigrammi di parole, lemmi e POS	0.75
2	Unigrammi, bigrammi, trigrammi di parole, bigrammi e trigrammi di POS e pentagrammi di caratteri	0.76
3	Unigrammi e bigrammi di lemmi, pentagrammi di caratteri	0.77
4	Bigrammi e trigrammi di POS, unigrammi e bigrammi di lemmi, pentagrammi di caratteri	0.77
5	Unigrammi, bigrammi, trigrammi di parole, bigrammi e trigrammi di POS e pentagrammi di caratteri	0.77
6	Unigrammi, bigrammi, trigrammi, tetragrammi di parole, bigrammi, trigrammi e tetragrammi di POS, pentagrammi e ottogrammi di caratteri	0.77
7	Unigrammi, bigrammi di parole, bigrammi, trigrammi e tetragrammi di POS, trigrammi di lemma e pentagrammi e ottogrammi di caratteri	0.77
8	Unigrammi, bigrammi, trigrammi di lemmi, trigrammi di lemma e pentagrammi di caratteri	0.78

Table 6: Descrizione degli esperimenti con Accuracy Media

Nella tabella seguente si riportano le metriche di valutazione del modello ottenuto dall'esperimento numero 8 sul test set, con iperparametri (C=0.1, dual=True, max_iter=25000) e, successivamente, l'accuracy sulle fold durante la k-fold cross-validation. Come prima, verifichiamo poi le 15 features che contribuiscono maggiormente alla classificazione.

Class	Precision	Recall	F1-Score	Support
0	0.75	0.75	0.75	641
1	0.74	0.75	0.74	621
Accuracy			0.75	1262
Macro Avg	0.75	0.75	0.75	1262
Weighted Avg	0.75	0.75	0.75	1262

Table 7: Esperimento 8: Performance sul test set

Fold	Accuracy
1	0.79
2	0.79
3	0.76
4	0.78
5	0.77
Mean Accuracy	0.78
Standard Deviation	0.01

Table 8: Esperimento 8: Accuracy per ogni fold e accuracy media

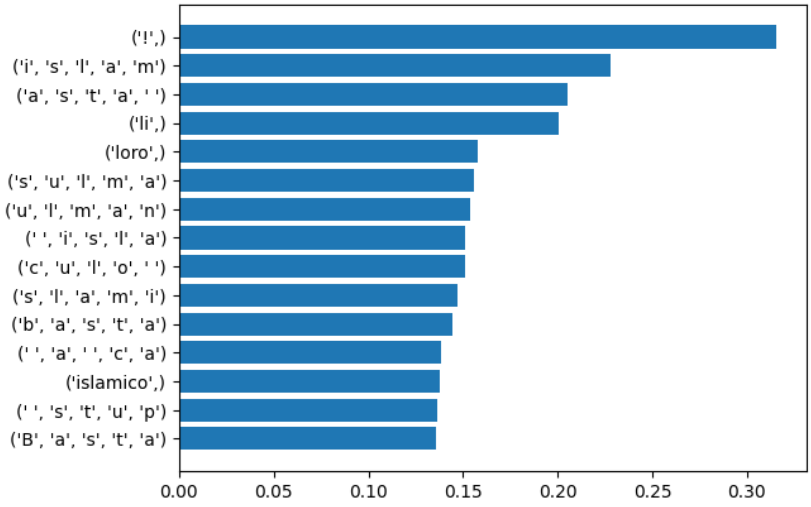

Figure 2: 15 features più importanti per la classificazione basata su features lessicali

Guardando l'immagine si nota come le feature più rilevanti siano per lo più ngrammi di caratteri. Interessante (ed estremamente ragionevole) la presenza, in prima posizione, del punto esclamativo, e poi di parole (ricomposte partendo dai caratteri) come *islam*, *musulmani* e *loro* e *li*. Queste ultime in particolare sarebbero un ottimo spunto per ulteriori riflessioni sociali e/o sociologiche. Con

un'accuracy di 0.75, il modello basato su ngrammi performa meglio del modello basato su feature linguistiche non lessicali.

3.3 Modello basato su word embeddings

Questo modello si allontana dai modelli visti precedentemente ed impiega rappresentazioni del testo basate sui Twitter Word Embeddings con 128 dimensioni, ottenuti con word2vec[9] partendo da un corpus di 46.935.207 tweet. Dopo aver ottenuto gli embedding delle parole, le feature sono state scalate con MinMaxScaler e sono stati condotti diversi esperimenti, creando modelli tra loro differenti per la natura delle parole prese in considerazione e la modalità di aggregazione degli embedding. Si riporta di seguito una tabella che riassume i suddetti esperimenti. Come prima, i risultati riportati nella colonna *Accuracy* fanno riferimento all'accuracy media ottenuta nel processo di kfold cross validation sul training set dal modello, con iperparametri selezionati attraverso GridSearch (combinato con kfold cross validation su 5 fold). Ciò che emerge è che le prestazioni migliori sono quelle di modelli che utilizzano tutte le parole, aggregandole tramite media non separate o somma (modelli che utilizzano, quindi, una rappresentazione costituita da un unico vettore, non derivato da concatenazione).

	Features	**Acc. Media**
1	Parole piene[10] e media separata[11]	0.73
2	Media non separata di tutte le parole	0.76
3	Media pesata con TFIDF[12] di tutte le parole	0.74
4	Somma di vettori di parole piene e nomi propri normalizzati con L2 norm	0.75
5	Somma di tutti i vettori normalizzati con L2 norm	0.76
6	Media separata di aggettivi e nomi	0.71
7	Media delle parole escluse le stopwords[13] [14]	0.76

Table 9: Descrizione degli esperimenti con Accuracy Media

Di seguito, si riportano i risultati della kfold cross validation e la prestazione del modello selezionato (Esperimento 7) sul test set.

[9] Rete neurale artificiale a due strati che ha lo scopo di apprendere rappresentazioni vettoriali di parole (gli embedding) che consentono di rappresentare una parola come un vettore in uno spazio vettoriale

Fold	Accuracy
1	0.76
2	0.77
3	0.75
4	0.75
5	0.76
Mean Accuracy	0.76
Standard Deviation	0.01

Table 10: Esperimento 7: Accuracy per ogni fold e accuracy media

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.73	0.76	0.74	641
1	0.74	0.71	0.72	621
Accuracy			0.73	1262
Macro Avg	0.73	0.73	0.73	1262
Weighted Avg	0.73	0.73	0.73	1262

Table 11: Esperimento 7: Performance sul test set

Di seguito i risultati dell'Esperimento 5 (con uguale accuracy).

Fold	Accuracy
1	0.76
2	0.77
3	0.75
4	0.75
5	0.76
Mean Accuracy	0.76
Standard Deviation	0.01

Table 12: Esperimento 5: Accuracy per ogni fold e accuracy media

Class	**Precision**	**Recall**	**F1-Score**	**Support**
0	0.74	0.73	0.74	641
1	0.73	0.74	0.73	621
Accuracy			0.73	1262
Macro Avg	0.73	0.73	0.73	1262
Weighted Avg	0.73	0.73	0.73	1262

Table 13: Esperimento 5: Performance sul test set

3.4 Neural Language Model

In questa sezione si esaminano l'implementazione e le prestazioni di un modello BERT[15], selezionato specificatamente per questo task. Nel contesto di questo studio, è stato utilizzato un modello BERT pre-addestrato per l'elaborazione di testi in lingua italiana, creato appositamente per il dominio specifico dei tweets. Il modello pre-addestrato impiegato è "m-polignano-uniba/bert_uncased_L-12_H-768_A-12_italian_alb3rt0". La scelta è ricaduta su tale modello, oltre che per la specificità del dominio, per la precisione e accuratezza della pipeline di preprocessing effettuata sui testi attraverso il tokenizer fornito[16]. Il modello è stato fine-tunato per 5 epoche e le metriche di valutazione sono state registrate per ogni epoca. È stata utilizzata una learning rate inziale di 2e-5, una batch size di 8 e una weight decay di 0.01. Al termine dell'addestramento è stato salvato il modello migliore. I testi sono stati tokenizzati utilizzando il tokenizzatore BERT con padding, troncamento e una lunghezza massima di 512 token. I dati sono stati preparati in formato torch per l'addestramento. Di seguito si riporta l'andamento della curva di loss e l'accuracy nel corso delle 5 epoche, e la performance del classificatore sul test set.

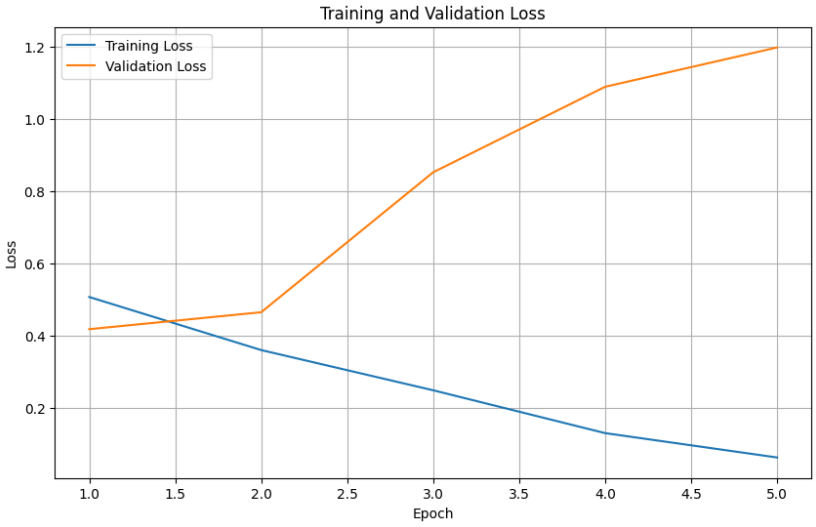

Figure 3: Andamento della loss tu training e validation data

[15] *Bidirectional Encoder Representations from Transformers*. Modello di deep learning sviluppato da Google, pre-addestrato su una vasta quantità di testi e addestrato utilizzando un'architettura di Transformer. La sua potenza è legata alla capacità di comprendere il contesto bidirezionale delle parole.

[16] Per maggiori dettagli si rimanda a https://github.com/marcopoli/AlBERTo-it/blob/master/tokenizer.py

Class	Precision	Recall	F1-Score	Support
0	0.76	0.81	0.78	641
1	0.79	0.74	0.76	621
Accuracy			0.77	1262
Macro Avg	0.77	0.77	0.77	1262
Weighted Avg	0.77	0.77	0.77	1262

Table 14: Performance sul test set

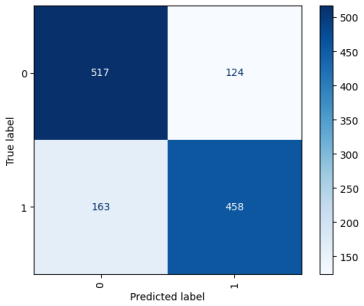

Figure 4: Confusion Matrix su Test Set

Come è possibile osservare, le curve di loss non sembrano convergere e, subito dopo la prima epoca, la loss sul validation set inizia ad aumentare, mentre quella di training diminuisce ad ogni epoca. Questo comportamento sembra suggerire una situazione di overfitting che, nel caso di un modello con milioni di parametri, sembra verosimile (e frequente) su dataset relativamente piccoli.

4 Conclusioni

Alla luce dei modelli testati e degli esperimenti condotti, è possibile asserire che, ad esclusione del modello basato su feature linguistiche non lessicali (con performance peggiori), gli altri modelli basati su Support Vector Machine lineari sembrano performare in maniera simile e, tutto sommato soddisfacente. Nell'ambito di ricerche future, ipotizzo che alcuni benefici potrebbero derivare dalla combinazione di feature linguistiche estratte con Profiling-Ud e feature lessicali basate sul numero di occorrenze di parole, ngrammi di parole e/o di caratteri. Ulteriori benefici potrebbero forse derivare anche dall'utilizzo di tecniche di preprocessing e tokenizzazione più complesse che porterebbero a un minor numero di token "mal formati" (favorendo il match con i dizionari di embedding forniti su italianlp.it), considerando che i testi scritti dagli utenti sui social network rappresentano, in questo senso, una sfida. Il modello basato su BERT ha prodotto un'accuracy di 0.77 su test set, risultando il modello migliore per il task proposto, seguito dal modello basato su ngrams (0.75) e da quelli basati su embeddings (0.73).

References

Brunato, D., A. Cimino, F. Dell'Orletta, S. Montemagni, and G. Venturi (2020, May). Profiling-ud: a tool for linguistic profiling of texts. pp. 11–16.

Cimino, A., L. De Mattei, and F. Dell'Orletta (2018, December). Multi-task learning in deep neural networks at evalita 2018. In *Proceedings of EVALITA '18, Evaluation of NLP and Speech Tools for Italian*, Turin, Italy.