Corso di Laurea Magistrale in Informatica Umanistica

Hate Speech and Stereotype Detection

Task di EVALITA 2020

Esame di Linguistica Computazionale II

Professori
Felice Dell'Orletta
Simonetta Montemagni
Giulia Venturi

Studente
Vincenzo Sammartino
Matricola 599203

A.A. 2022/2023

Indice

1 Introduzione

In questo report vengono mostrati i risultati ottenuti dal confronto di vari modelli per svolgere i vari task di *HaSpeeDe (Hate Speech Detection Task)*[3], un task di classificazione di odio e stereotipi proposto da EVALITA[1] nel 2020.

Nella sezione successiva vengono brevemente descritti i dataset usati per svolgere i task e il preprocessing applicato ad essi.

Nella sezione 3 viene illustrato il modello di *machine learning* usato per la classificazione dei testi, mentre nelle sottosezioni vengono spiegati i vari modelli per la vettorizzazione del testo con i risultati degli esperimenti.

2 Dataset

Per svolgere i task sono stati usati diversi dataset in formato TSV (Tab Separated Value)[2] le cui distribuzioni sono mostrate nelle tabelle 1 e 2.

Dataset	HS	Non HS	Totale
Training Set	2766	4073	6839
Test Set News	181	319	500
Test Set Tweet	622	641	1263

Tabella 1: Distribuzione dell'Hate Speech nel Training Set e nel Test Set

Dataset	Stereotipo	Non Stereotipo	Totale
Training Set	3042	3797	6839
Test Set News	175	325	500
Test Set Tweet	569	694	1263

Tabella 2: Distribuzione degli Stereotipi nel Training Set e nel Test Set

I dati sono formattati nel seguente modo:

```
id text hs stereotype
```

1. Un numero univoco che sostituisce l'ID originale del tweet
2. Il testo del tweet
3. La classe *Hate Speech*: 1 se il testo contiene odio, 0 altrimenti
4. La classe *Stereotype*: 1 se il testo contiene stereotipi, 0 altrimenti

I dataset in formato TSV sono stati trasformati in txt che contiene tweet/news divisi per riga.

Un piccolo preprocessing è stato effettuato rimuovendo le occorrenze di "@user" e "URL" in modo da ottenere dei testi più puliti.

Per trasformare le features (X) è stato usato lo scaler `MaxAbsScaler`[3].

In questa relazione, per motivi di brevità, vengono mostrati solo i risultati ottenuti nel dataset dei Tweet; infatti le caratteristiche emerse dagli esperimenti su questo dataset sono molto simili a quelle ottenute dagli esperimenti effettuati sul dataset delle news (per approfondire si può far riferimento ai due notebook della classificazione delle news).

[1] EVALITA è un'iniziativa che mira a promuovere lo sviluppo e la diffusione di risorse e tecnologie per il trattamento automatico della lingua italiana, attraverso l'organizzazione di competizioni e la condivisione di risorse linguistiche e strumenti.

[2] TSV: Tab Separated Value, un formato in cui ogni campo è separato da un carattere di tabulazione.

[3] Il `MaxAbsScaler` è un metodo di pre-elaborazione dei dati in scikit-learn che scala ogni caratteristica dividendo per il valore assoluto massimo di quella caratteristica. Questo processo porta le caratteristiche nell'intervallo $[-1, 1]$. È particolarmente utile quando si lavora con dati sparsi, poiché mantiene la struttura sparsa dei dati.

3 Modelli

Tutti i risultati degli esperimenti sono elencati nelle seguenti sottosezioni raggruppate per modello di rappresentazione utilizzato.

La *baseline* usata come riferimento è stata fatta con un `Dummy Classifier`[4] nella sua strategia `prior`, etichettando tutti i documenti con la classe più frequente, etichettando tutti i documenti come "non contenenti hate-speech" (0) e "non contenenti stereotipi" (0).

Per tutti i modelli, ad esclusione della sezione 3.7 in cui è stata usato un Neural Language Model, è stato usato il modello di classificazione `LinearSVC`: si tratta di un'implementazione del classificatore *Support Vector Machine* (SVM) basata sulla libreria `liblinear`, che è progettata per lavorare con kernel lineari e adattarsi meglio a un gran numero di campioni rispetto alla libreria `libsvm`. Questo modello supporta sia input densi[5] (ad esempio Word Embeddings) che sparsi[6] (modelli come Bag of Words) e gestisce il supporto multiclasse secondo uno schema *"one-vs-rest"* (ovr).

In questi esperimenti si è scelto di utilizzare due `LinearSVC` separati per la classificazione binaria di odio e stereotipi, senza fare una classificazione multiclasse del tipo `[hs, stereotype]`, date le migliori prestazioni ottenute dai singoli classificatori binari.

Per il modello `LinearSVC` è stato effettuato il tuning degli iperparametri; in particolare si ottengono notevoli variazioni di performance al variare di C[7], come si può vedere in figura 1:

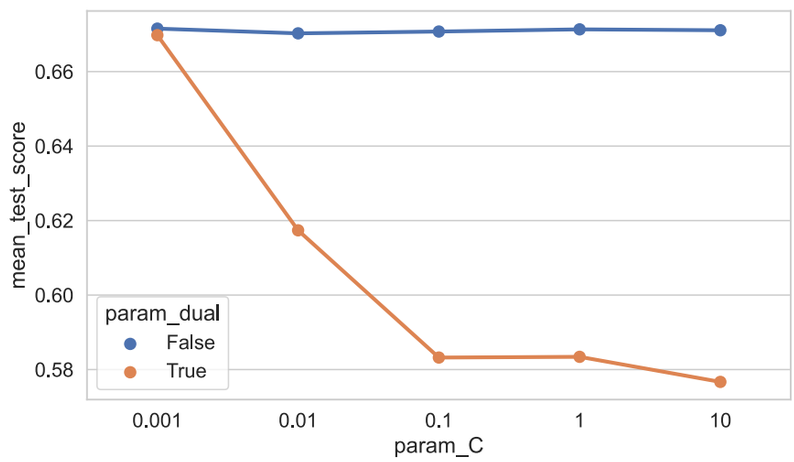

Figura 1: Accuratezza al variare del parametro C di LinearSVC

Una volta effettuato il tuning degli iperparametri sul training set, si è scelto di usare la seguente configurazione per tutti i test effettuati:

```
LinearSVC(C=0.001, dual=False)
```

3.1 Informazioni linguistiche non lessicali

I primi test sono stati effettuati usando le informazioni linguistiche non lessicali estratte con il sistema ProfilingUD [1], uno strumento per il profiling linguistico dei testi.

[4] Un Dummy Classifier è utilizzato principalmente come punto di riferimento per confrontare i risultati ottenuti da algoritmi di apprendimento automatico più avanzati. Nonostante la sua semplicità, può essere utile in situazioni in cui è necessario un risultato rapido o per fornire una baseline per confrontare le prestazioni di altri modelli.

[5] Una matrice densa è una matrice in cui la maggior parte degli elementi è diversa da zero. A differenza delle matrici sparse, le matrici dense non beneficiano di algoritmi e strutture dati ottimizzate per la gestione della sparsità.

[6] Una matrice sparsa è una matrice i cui valori sono quasi tutti uguali a zero. Si collega ai sistemi accoppiati e risulta proficuo utilizzare algoritmi specializzati e strutture dati per gestirle in modo efficiente. *(Fonte: [it.wikipedia.org](https://it.wikipedia.org/wiki/Matrice_sparsa))*

[7] Il valore di C di LinearSVM è un parametro di regolazione che determina la costante di smoothing nella formula di classificazione. Un valore di C più alto rende il modello più permissivo, consentendo una maggiore variazione nella classificazione, mentre un valore di C più basso rende il modello più rigido, riducendo la variazione nella classificazione.

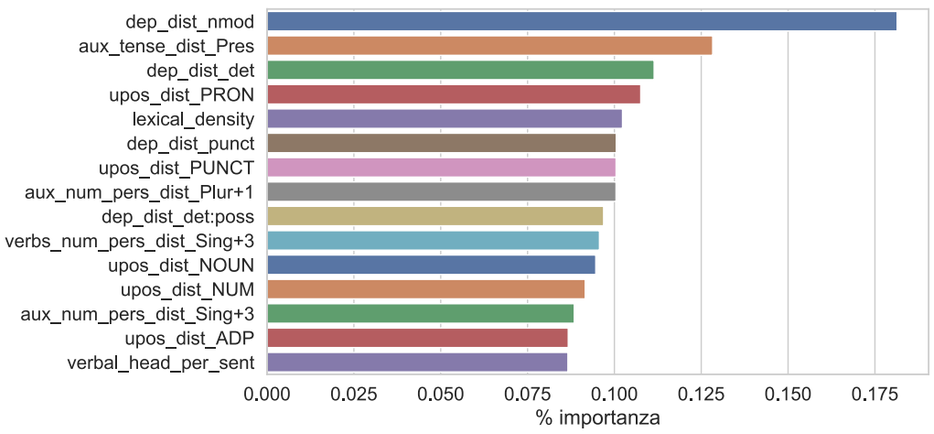

Figura 2: Lista delle 15 features più importanti per la classificazione

La figura 2 mostra l'importanza relativa di diverse caratteristiche linguistiche e sintattiche analizzate in questo studio. Si osserva che la caratteristica `dep_dist_nmod` (distanza di dipendenza tra modificatori nominali) ha la maggiore importanza (0.181395), seguita da `aux_tense_dist_Pres` (distanza tra ausiliari al tempo presente) (0.128296) e `dep_dist_det` (distanza di dipendenza tra determinanti) (0.111437).

È interessante notare che alcune caratteristiche, come `verbal_head_per_sent` (numero medio di teste verbali per frase) (0.086617) e `upos_dist_ADP` (distribuzione delle parti del discorso per le preposizioni) (0.086711), hanno un'importanza relativamente inferiore in questa analisi.

Task	**5-fold Cross Validation**				
HS	0.6364	0.5929	0.5455	0.5675	0.6032
	Media: 0.5891				
Stereotype	0.5692	0.5968	0.5573	0.5675	0.5516
	Media: 0.5685				

Tabella 3: Risultati della 5-fold cross validation per i due task

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.60	0.61	0.60	0.60
HS		0.59	0.61	0.60
Non Stereotype	0.57	0.60	0.67	0.63
Stereotype		0.53	0.44	0.48

Tabella 4: Risultati della classificazione con le informazioni linguistiche non lessicali

I risultati mostrano che le informazioni linguistiche non lessicali producono performance simili nei due task; nella classificazione dell'odio l'accuratezza è leggermente migliore. Il rilevamento degli stereotipi risulta essere più complesso, in particolare i valori di precision, recall e di conseguenza f1-score sono piuttosto bassi.

3.2 Bag of Words

In questa sezione vengono mostrati i risultati ottenuti con l'uso del modello Bag of Words, una delle tecniche più semplici e intuitive per convertire il testo in formato numerico. L'idea di base è creare un vettore di parole presenti in un corpus di testi e contare la frequenza di ogni parola in un documento specifico. Per illustrare il funzionamento del modello BoW, consideriamo il seguente esempio:

Supponiamo di avere i seguenti documenti:

- Documento 1: "Il gatto gioca nel giardino."
- Documento 2: "Il cane corre nel parco."

Per trasformare il testo in vettori, vengono eseguiti i seguenti passaggi:

1. Creare un vocabolario con tutte le parole univoche presenti nei documenti:

$$Vocabolario = \{Il, gatto, gioca, nel, giardino, cane, corre, parco\}$$

2. Per ogni documento, creare un vettore di frequenza delle parole basato sul vocabolario:

- Documento 1: $[1, 1, 1, 1, 1, 0, 0, 0]$
- Documento 2: $[1, 0, 0, 1, 0, 1, 1, 1]$

Una volta trasformato il testo in vettori, questi ultimi vengono sottoposti al classificatore sotto forma di matrice sparsa (`X_train/X_test`).

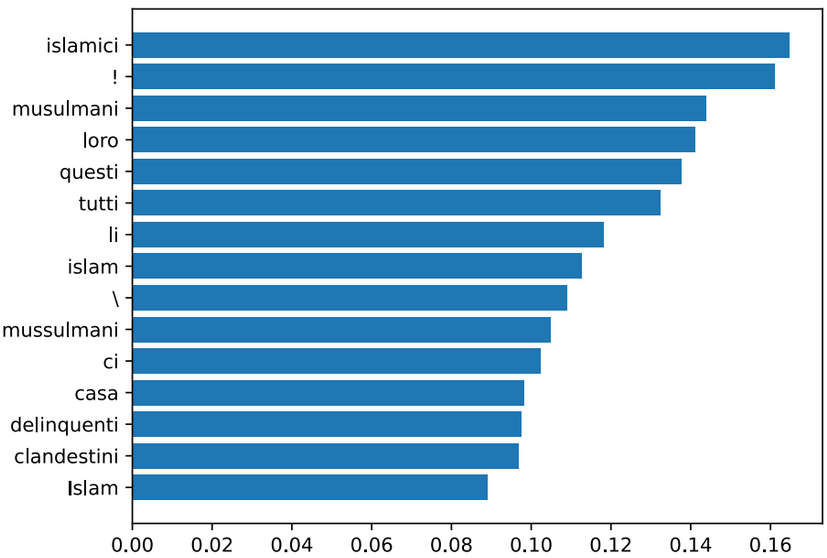

Figura 3: Lista delle 15 features più importanti per la classificazione (forme)

Nella figura 3 vengono mostrate le 15 features che risultano essere più importanti per il classificatore, ordinate per la loro importanza in termini percentuali. Tra queste ci sono molte parole che potrebbero essere associate sia a discorsi di odio che per la classificazione degli stereotipi.

In questo contesto potrebbe essere naturale pensare che basterebbe realizzare una "lista predefinita" di termini noti solitamente associati a discorsi d'odio (es. islamici, musulmani, delinquenti, clandestini ecc.). In realtà è importante notare che la presenza di queste parole da sole non implica necessariamente un discorso d'odio; infatti il contesto in cui queste parole vengono utilizzate è fondamentale per determinare se si tratta effettivamente di un discorso d'odio o meno. Per questo è meglio affidarsi a modelli di apprendimento automatico come quelli riportati in questa relazione sottoponendo anche il contesto delle parole (ad esempio i modelli che usano gli n-grammi di parole).

Task	**5-fold Cross Validation**				
HS	0.7036	0.6838	0.5968	0.5833	0.6270
	Media: 0.6389				
Stereotype	0.6442	0.6245	0.5731	0.6547	0.5873
	Media: 0.6167				

Tabella 5: Risultati della 5-fold cross validation per i due task

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.69	0.69	0.73	0.71
HS		0.70	0.66	0.68
Non Stereotype	0.65	0.69	0.66	0.68
Stereotype		0.61	0.65	0.63

Tabella 6: Risultati della classificazione con il modello BoW sulle forme di parole

Queste tabelle mostrano che il modello BoW sulle forme di parole ha prestazioni nettamente migliori rispetto al modello basato su informazioni linguistiche non lessicali in entrambi i task.

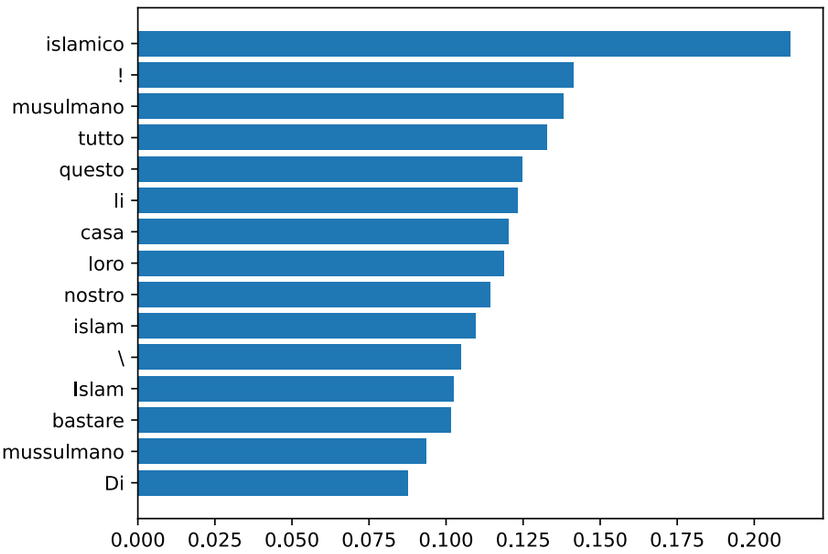

Figura 4: Lista delle 15 features più importanti per la classificazione (lemmi)

Nella figura 4 è evidente come la feature "islamico" abbia un peso molto maggiore rispetto a tutte le altre, fenomeno che non è così evidente nella classificazione con le forme (figura 3); va comunque sottolineato che molte features sono comuni al modello che usa le forme.

Task	**5-fold Cross Validation**				
HS	0.7589	0.6877	0.6245	0.6111	0.6429
	Media: 0.6650				
Stereotype	0.6759	0.6245	0.5692	0.6627	0.5992
	Media: 0.6263				

Tabella 7: Risultati della 5-fold cross validation per i due task

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.70	0.69	0.75	0.72
HS		0.72	0.65	0.68
Non Stereotype	0.67	0.71	0.69	0.70
Stereotype		0.63	0.65	0.64

Tabella 8: Risultati della classificazione con il modello BoW sui lemmi di parole

L'uso dei lemmi al posto delle parole nel modello Bag Of Words porta un leggero incremento di prestazioni per entrambi i task.

3.3 N-grammi di caratteri

Il modello Bag of n-grams è un'estensione del modello Bag of Words e consiste nel creare vettori di frequenza basati su sequenze di n caratteri anziché parole singole. Gli n-grammi di caratteri possono catturare maggiormente il contesto delle parole e aiutare a riconoscere pattern e similitudini tra diversi documenti.

Negli esperimenti condotti per questo report sono stati usati gli n-grammi di caratteri fino a 6 caratteri. Infatti, pur aumentando il valore dei caratteri non si ottiene un miglioramento delle prestazioni, mentre si ottiene un notevole aumento del numero degli elementi inseriti nel vocabolario, caratteristica che porta a un aumento di memoria occupata non indifferente.

Di seguito vengono mostrati i passaggi effettuati per creare i vettori contenenti gli n-grammi di caratteri con $n = 2$:

1. Creare un insieme di n-grammi di caratteri con $n = 2$ per entrambi i documenti:
   - Documento 1: {Il, l_, _g, ga, at, tt, to, o_, _g, gi, io, oc, ca}
   - Documento 2: {Il, l_, _c, ca, an, ne, e_, _c, co, or, re}
2. Per ogni documento, creare un vettore di frequenza degli n-grammi di caratteri basato sull'unione degli insiemi di n-grammi di caratteri:

- Vocabolario: {Il, l_, _g, ga, at, tt, to, o_, _g, gi, io, oc, ca, _c, co, or, re, an, ne, e_}
- Documento 1: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]
- Documento 2: [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]

Come si può vedere da questo brevissimo esempio, il numero di elementi inseriti nel vocabolario inizia a diventare molto grande. Infatti, per ridurre il numero di elementi nel vocabolario è necessario rimuovere gli elementi che occorrono meno di n volte (nello specifico caso di questo studio sono state rimosse le occorrenze inferiori a 5).

I vettori risultanti da questo processo di trasformazione vengono dati in input al classificatore e di seguito vengono mostrate le 15 features più importanti per la classificazione:

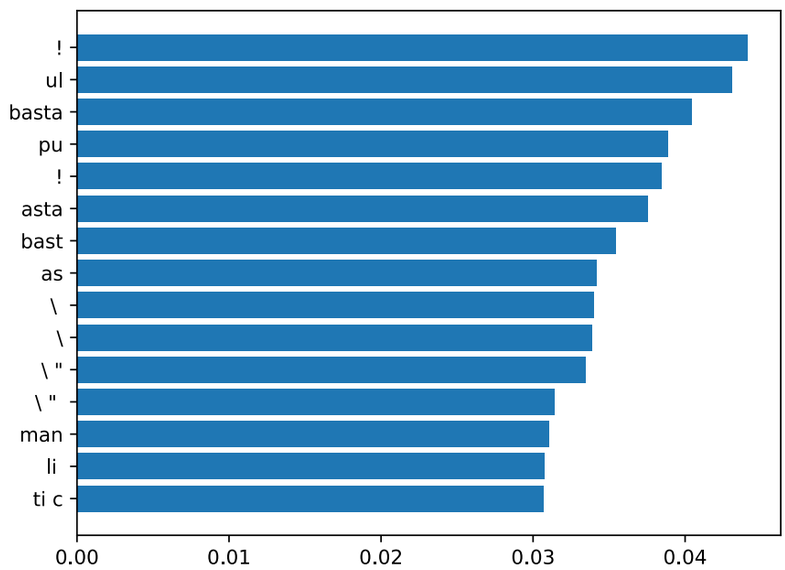

Figura 5: Lista delle 15 features più importanti per la classificazione

Ciò che si può notare dalla figura 5 è che il peso assegnato alle features più importanti è basso in relazione agli altri modelli visti in questo report: ciò è dovuto anche alla grandezza del vocabolario che è nettamente maggiore rispetto agli altri modelli.

Task	**5-fold Cross Validation**				
HS	0.7115	0.7549	0.6403	0.6865	0.7500
	Media: 0.7086				
Stereotype	0.7075	0.7154	0.6482	0.7341	0.7103
	Media: 0.7031				

Tabella 9: Risultati della 5-fold cross validation per i due task

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.74	0.75	0.74	0.75
HS		0.74	0.75	0.74
Non Stereotype	0.72	0.77	0.69	0.73
Stereotype		0.66	0.75	0.71

Tabella 10: Risultati della classificazione con il modello che usa gli n-grammi di caratteri

Le tabelle mostrano che il modello che usa gli n-grammi di caratteri ottengono ottimi risultati; infatti per la classificazione degli stereotipi si tratta del modello che ha ottenuto i risultati migliori.

3.4 N-grammi di parole

I modelli con n-grammi di parole possono venirci in aiuto per riuscire a catturare il contesto delle parole: all'interno del vocabolario non vengono inserite solo le singole parole, bensì n-grammi di parole fino a un valore predefinito di n.

Nel caso specifico di questo studio si è deciso di usare gli n-grammi fino a 6, dato il numero basso di elementi contenuti nel vocabolario. Avendo dataset più grandi non sarebbe possibile raggiungere questo valore a causa della notevole occupazione di memoria.

Per rappresentare documenti, creiamo un insieme di n-grammi di parole con n=2 per entrambi i documenti:

- Documento 1: {"Mi piace", "piace ballare", "ballare sotto", "sotto la", "la pioggia"}
- Documento 2: {"Mi piace", "piace correre", "correre nel", "nel parco"}

Per ogni documento, creiamo un vettore di frequenza degli n-grammi di parole basato sull'unione degli insiemi di n-grammi di parole:

- Vocabolario n-grammi di parole: {"Mi piace", "piace ballare", "ballare sotto", "sotto la", "la pioggia", "piace correre", "correre nel", "nel parco"}
- Documento 1: [1, 1, 1, 1, 1, 0, 0, 0]
- Documento 2: [1, 0, 0, 0, 0, 1, 1, 1]

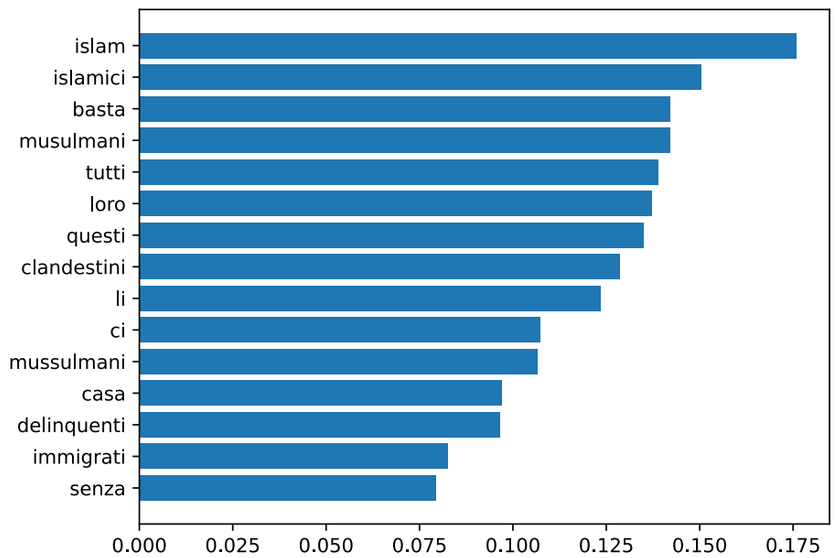

Figura 6: Lista delle 15 features più importanti per la classificazione

In questa figura un'importante caratteristica che si può osservare è che gli elementi che vengono presi maggiormente in considerazione sono comunque le singole parole: gli n-grammi di parole hanno infatti un peso percentuale maggiore rispetto agli n-grammi. Va comunque tenuto in considerazione che il peso delle singole parole è nettamente inferiore a quello ottenuto usando solo le singole parole (figura 3).

Task	**5-fold Cross Validation**				
HS	0.6917	0.7115	0.5375	0.6429	0.6786
	Media: 0.6524				
Stereotype	0.6403	0.6680	0.5889	0.6548	0.6706
	Media: 0.6445				

Tabella 11: Risultati della 5-fold cross validation per i due task

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.70	0.69	0.76	0.72
HS		0.72	0.64	0.68
Non Stereotype	0.69	0.74	0.66	0.70
Stereotype		0.64	0.72	0.68

Tabella 12: Risultati della classificazione con il modello che usa gli n-grammi di caratteri

3.5 N-grammi di PoS

In questa sezione vengono mostrati i risultati dei test effettuati con l'uso delle Part-of-Speech (PoS) piuttosto che le singole parole; la vettorizzazione è stata effettuata seguendo i seguenti passaggi:

1. Effettuare il PoS tagging per entrambi i documenti:
   - Documento 1: {(Il, DT), (gatto, NN), (gioca, VB), (con, IN), (la, DT), (palla, NN), (nel, IN), (giardino, NN)}
   - Documento 2: {(Il, DT), (cane, NN), (corre, VB), (veloce, RB), (nel, IN), (parco, NN), (e, CC), (abbaia, VB)}
2. Creare un insieme di n-grammi di POS con $n = 2$ per entrambi i documenti:
   - Documento 1: {(DT, NN), (NN, VB), (VB, IN), (IN, DT), (DT, NN), (NN, IN), (IN, NN)}
   - Documento 2: {(DT, NN), (NN, VB), (VB, RB), (RB, IN), (IN, NN), (NN, CC), (CC, VB)}
3. Per ogni documento, creare un vettore di frequenza degli n-grammi di POS basato sull'unione degli insiemi di n-grammi di POS:
   - Vocabolario n-grammi di POS: {(DT, NN), (NN, VB), (VB, IN), (IN, DT), (DT, NN), (NN, IN), (IN, NN), (VB, RB), (RB, IN), (NN, CC), (CC, VB)}
   - Documento 1: $[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]$
   - Documento 2: $[1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1]$

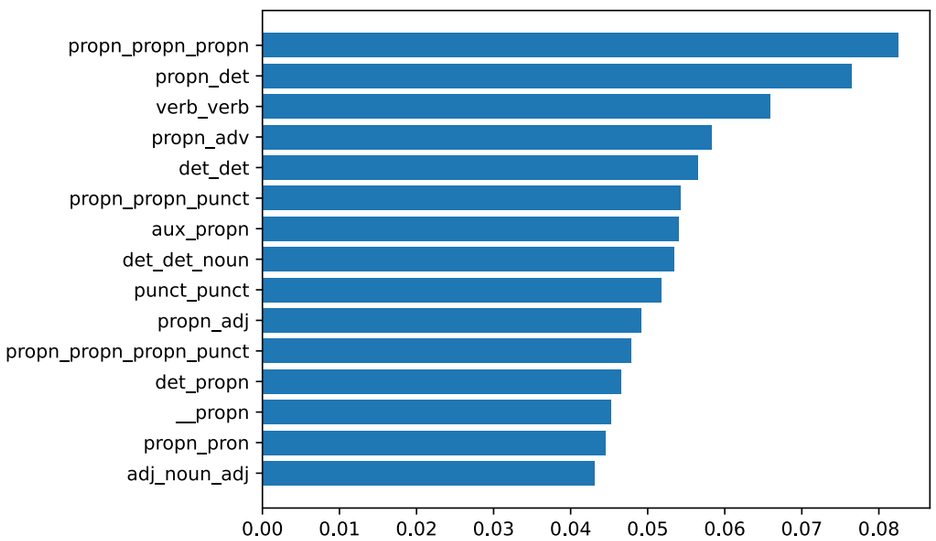

Figura 7: Lista delle 15 features più importanti per la classificazione

Una cosa interessante che si può notare in questa analisi è che il peso attribuito alle features più importanti è nettamente più basso rispetto ai modelli che presentano molte più features (come Bag of Words di singole parole).

Task	**5-fold Cross Validation**				
HS	0.6917	0.7115	0.5375	0.6429	0.6786
	Media: 0.6524				
Stereotype	0.6403	0.6680	0.5889	0.6548	0.6706
	Media: 0.6445				

Tabella 13: Risultati della 5-fold cross validation per i due task

Metrica	Accuracy	Precision	Recall	F1-score
Non HS	0.60	0.60	0.65	0.62
HS		0.61	0.55	0.58
Non Stereotype	0.59	0.63	0.59	0.61
Stereotype		0.54	0.59	0.56

Tabella 14: Risultati della classificazione con il modello che usa gli n-grammi di PoS

I risultati mostrano che usando le PoS non si ottengono buone prestazioni per entrambi i task; infatti per la classificazione degli stereotipi è il modello che ottiene i risultati peggiori (pur restando al di sopra della baseline).

3.6 Word Embeddings

Il modello Word Embeddings pone un approccio totalmente differente dai modelli visti precedentemente, infatti si parla di "rappresentazione implicita del linguaggio": le singole parole vengono rappresentate attraverso un vettore di n elementi che rappresenta le sue caratteristiche semantiche.

La struttura dati che si viene a creare con questo modello è una matrice di tipo denso, che è contrapposta alle matrici sparse prodotte dai modelli precedenti.

In questi esperimenti sono stati usati sia gli embeddings itWac che quelli fatti su Twitter forniti da ItaliaNLP Lab [2].

Le strategie seguite per creare i vettori sono state differenti:

- Un vettore di 128 elementi risultante dalla media di tutti gli embeddings della frase
- Un vettore di 384 elementi (128 + 128 + 128) risultante dalla media di sostantivi, aggettivi e verbi.

Metrica	Accuracy	Precision	Recall	F1-score
Non HS	0.70	0.69	0.73	0.71
HS		0.71	0.67	0.69
Non Stereotype	0.67	0.71	0.67	0.69
Stereotype		0.62	0.67	0.65

Tabella 15: Risultati della classificazione con il modello Word Embedding (itWac) usando un vettore singolo

Task	5-fold Cross Validation				
HS	0.7036	0.7233	0.6285	0.6944	0.7540
Media: 0.7007					
Stereotype	0.7115	0.6324	0.5731	0.6905	0.6667
Media: 0.6445					

Tabella 16: Risultati della 5-fold cross validation per i due task (itWac, singolo vettore)

Seguendo la strategia del vettore singolo di 128 elementi, si ottengono dei buoni risultati per entrambi i task, leggermente migliori per la classificazione dell'odio.

Metrica	Accuracy	Precision	Recall	F1-score
Non HS	0.63	0.62	0.69	0.65
HS		0.64	0.56	0.60
Non Stereotype	0.64	0.67	0.69	0.68
Stereotype		0.60	0.58	0.59

Tabella 17: Risultati della classificazione con il modello Word Embedding (itWac) usando 3 vettori

Task	5-fold Cross Validation				
HS	0.7036	0.7233	0.6285	0.6944	0.7540
Media: 0.7007					
Stereotype	0.7115	0.6324	0.5731	0.6905	0.6667
Media: 0.6445					

Tabella 18: Risultati della 5-fold cross validation per i due task (itWac, 3 vettori)

Nella tabella 17 si può vedere che la strategia che usa 3 vettori separati contenenti la media delle "parole piene" (sostantivi, aggettivi e verbi) effettivamente non fa ottenere un miglioramento in termini di prestazioni di classificazione.

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.73	0.71	0.77	0.74
HS		0.74	0.68	0.71
Non Stereotype	0.67	0.70	0.71	0.70
Stereotype		0.64	0.63	0.63

Tabella 19: Risultati della classificazione con il modello Word Embedding (Twitter) usando un vettore singolo

Task	**5-fold Cross Validation**				
HS	0.7984	0.6957	0.6166	0.6667	0.6349
	Media: 0.6825				
Stereotype	0.6403	0.6008	0.5850	0.6389	0.5913
	Media: 0.6112				

Tabella 20: Risultati della 5-fold cross validation per i due task (Twitter, 1 vettore)

Attraverso l'uso di embeddings creati nello stesso dominio, si ottiene un buon miglioramento delle prestazioni nella classificazione dell'odio, mentre nella classificazione degli stereotipi le prestazioni rimangono invariate.

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.69	0.68	0.72	0.70
HS		0.70	0.65	0.67
Non Stereotype	0.65	0.67	0.71	0.69
Stereotype		0.62	0.58	0.60

Tabella 21: Risultati della classificazione con il modello Word Embedding (Twitter) usando 3 vettori

Task	**5-fold Cross Validation**				
HS	0.7984	0.6957	0.6166	0.6667	0.6349
	Media: 0.6825				
Stereotype	0.6719	0.6008	0.5850	0.6389	0.5913
	Media: 0.612				

Tabella 22: Risultati della 5-fold cross validation per i due task (Twitter, 3 vettori)

La strategia dei 3 vettori creati su Twitter porta a un notevole miglioramento nella classificazione dell'odio e un leggero miglioramento nella classificazione degli stereotipi, anche se la strategia che usa un singolo vettore con 128 elementi comunque ottiene migliori risultati.

3.7 Neural Language Model

L'esperimento più avanzato per questo studio è rappresentato dal modello "bert-base-italian-cased", un modello basato su BERT specifico per la lingua italiana.

Questo modello è stato pre-addestrato con una rete neurale a 12 strati e 768 dimensioni nascoste su un corpus italiano di circa 30 GB di testo, che includeva diversi tipi di testo, come notizie, libri, conversazioni, testi scientifici e altro ancora. Esso può essere utilizzato per fini di elaborazione del linguaggio naturale, come classificazione del testo, analisi del sentimento, risposta alle domande e generazione di testo.

In questi esperimenti è stato effettuato un fine-tuning di 5 epoche sul training set con una *batch size* pari a 80 (una dimensione così grande richiede circa 12 GB di RAM della GPU).

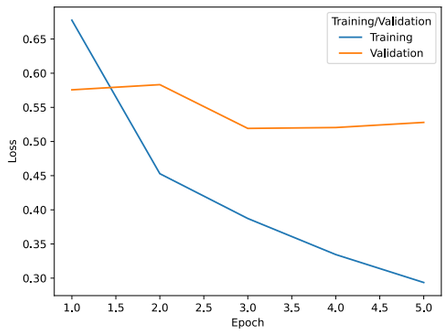

Figura 8: Andamento della loss function nelle 5 epoche per la HS Classification

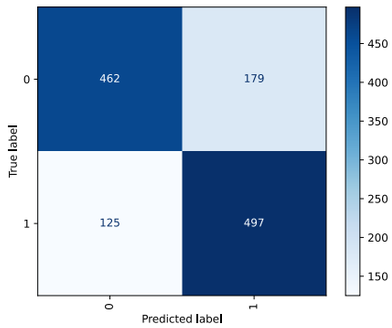

Figura 9: Confusion Matrix per le due classi di hate speech

Nella figura 8 viene mostrato un *lineplot* della loss, un grafico che rappresenta la funzione di perdita (loss function) di un algoritmo di apprendimento automatico durante l'addestramento. Questo grafico aiuta a visualizzare il processo di ottimizzazione dell'algoritmo e a comprendere se l'algoritmo sta imparando correttamente dai dati di addestramento. In altre parole, mostra come la perdita diminuisce man mano che l'algoritmo migliora nel corso delle iterazioni o delle epoche.

Nel caso della classificazione dell'odio l'andamento è decrescente per il training set, mentre nel caso del validation set la loss rimane costante.

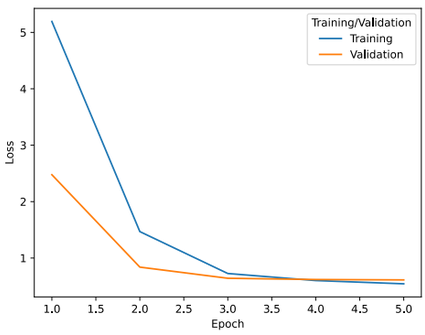

Figura 10: Andamento della loss function nelle 5 epoche nella classificazione degli stereotipi

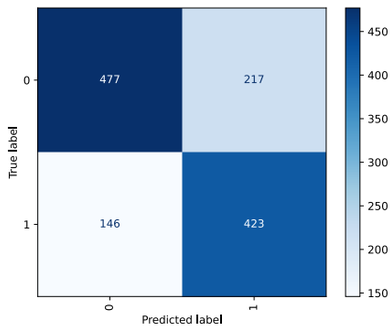

Figura 11: Confusion Matrix per le due classi degli stereotipi

Nel caso della classificazione degli stereotipi si vede un netto calo della loss, sia per il training che per il validation set nelle prime 3 epoche, per poi rimanere piuttosto costante nelle due successive.

Metrica	**Accuracy**	**Precision**	**Recall**	**F1-score**
Non HS	0.76	0.79	0.72	0.75
HS		0.74	0.80	0.77
Non Stereotype	0.71	0.77	0.69	0.72
Stereotype		0.66	0.74	0.70

Tabella 23: Risultati della classificazione con il Neural Language Model

La tabella 23 mostra che il modello ha una performance leggermente migliore nella classificazione di Hate Speech rispetto alla classificazione degli stereotipi.

Per la classificazione di Hate Speech il modello ha classificato correttamente il 76% degli esempi. La Precision è 0.79 per la classe Non HS e 0.74 per la classe HS, ciò significa che il modello ha una maggiore capacità di identificare correttamente gli esempi Non HS rispetto alla classe HS. Il Recall, invece, è più alto per la classe HS (0.80) rispetto alla classe Non HS (0.72), il che implica che il modello è più sensibile nel rilevare gli esempi di Hate Speech rispetto a quelli Non HS. Infine, l'F1-score, che combina Precision e Recall, è 0.75 per la classe Non HS e 0.77 per la classe HS, il che indica un bilanciamento generale tra Precision e Recall.

Per la classificazione degli stereotipi, l'Accuracy è 0.71, mostrando che il modello ha correttamente classificato il 71% degli esempi. La Precision è 0.77 per la classe Non Stereotype e 0.66 per la classe Stereotype: il modello ha una maggiore capacità di identificare correttamente gli esempi Non Stereotype rispetto alla classe Stereotype. Il Recall, invece, è più alto per la classe Stereotype (0.74) rispetto alla classe Non Stereotype (0.69), indicando che il modello è più sensibile nel rilevare gli esempi di Stereotype rispetto a quelli Non Stereotype. Infine, l'F1-score è 0.72 per la classe Non Stereotype e 0.70 per la classe Stereotype, ciò mostra un bilanciamento generale tra Precision e Recall.

Epoch	Training Loss	Validation Loss	F1 Score
1	0.677600	0.575667	0.730411
2	0.452900	0.583253	0.716739
3	0.387300	0.519184	0.759018
4	0.334500	0.520468	0.753589
5	0.293600	0.527987	0.763272

Tabella 24: Valori di loss e F1 score durante le 5 epoche di fine-tuning per la HS Classification

Nella tabella 24 si può osservare che la Training Loss diminuisce costantemente in ogni epoca, dimostrando che il modello sta imparando dai dati di addestramento.

La Validation Loss non mostra un andamento decrescente simile a quello della Training Loss e sembra oscillare durante le epoche. Questo potrebbe suggerire che il modello non migliora significativamente la sua capacità di generalizzare su nuovi dati.

Anche il punteggio F1 Score tende a oscillare, andando comunque ad aumentare, passando da 0.730411 a 0.763272. Ciò significa che il modello sta migliorando la sua capacità di classificare correttamente gli esempi di Hate Speech, ma c'è ancora margine di miglioramento.

Epoch	Training Loss	Validation Loss	F1 Score
1	5.193100	2.476944	0.389722
2	1.468800	0.839115	0.649148
3	0.725700	0.641396	0.698239
4	0.600900	0.619095	0.698847
5	0.543300	0.612417	0.713281

Tabella 25: Valori di loss e F1 score durante le 5 epoche di fine-tuning per la Stereotype Classification

Dalla tabella 25 emerge che la Training Loss diminuisce costantemente in ogni epoca, ciò indica che il modello sta imparando dai dati di addestramento. La Validation Loss non mostra lo stesso andamento decrescente, questo potrebbe suggerire un possibile overfitting del modello ai dati di addestramento.

Il punteggio F1 Score, che tiene conto sia della precisione che del recall, aumenta generalmente durante le epoche, passando da 0.389722 a 0.713281. Questo denota che il modello sta migliorando la sua capacità di classificare correttamente gli stereotipi, ma c'è ancora margine di miglioramento.

Il modello sta apprendendo dai dati di addestramento, ma potrebbe essere soggetto a overfitting. Per migliorare ulteriormente il modello, si potrebbero considerare tecniche di regolarizzazione, come l'utilizzo di dropout[8] o la riduzione della complessità del modello, nonché l'aggiustamento dei parametri di ottimizzazione come il learning rate[9].

[8]Il dropout prevede l'eliminazione casuale di alcuni nodi durante l'addestramento, riducendo la complessità del modello.

[9]Un learning rate più elevato può far convergere il modello più rapidamente, ma potrebbe causare instabilità e oscillazioni nella loss. Un learning rate più basso può portare a una convergenza più lenta, ma stabile.

4 Conclusioni

In questa sezione vengono riportati complessivamente i risultati ottenuti dagli esperimenti effettuati.

Modello	Accuracy	Precision	Recall	F1-score
Informazioni linguistiche non lessicali	0.60	0.60	0.60	0.60
Bag of Words (forme)	0.69	0.69	0.69	0.69
Bag of Words (lemmi)	0.70	0.70	0.70	0.70
N-grammi di caratteri	0.74	0.74	0.74	0.74
N-grammi di parole	0.70	0.70	0.70	0.70
N-grammi di POS	0.60	0.60	0.60	0.60
Word Embeddings itWac (1 vettore)	0.70	0.70	0.70	0.70
Word Embeddings itWac (3 vettori)	0.63	0.63	0.63	0.63
Word Embeddings Twitter (1 vettore)	0.73	0.73	0.72	0.72
Word Embeddings Twitter (3 vettori)	0.69	0.69	0.69	0.69
Word Embeddings (Word2Vec)	0.63	0.65	0.62	0.61
Neural Language Model	**0.76**	**0.76**	**0.76**	**0.76**
Baseline	0.51	0.25	0.50	0.34

Tabella 26: Metriche di valutazione per i diversi modelli nella classificazione dell'odio

La classificazione dell'odio risulta essere un task in cui i modelli performano meglio, pur essendo più bassa la baseline.

In entrambi i task la scelta di usare 3 vettori combinati (la strategia è spiegata nel dettaglio nella sezione 3.6) non produce migliori risultati.

Nel caso della classificazione dell'odio, i migliori risultati sono ottenuti dal Neural Language Model che ottiene un buon 76% di accuratezza.

Per entrambi i task il modello che usa gli n-grammi di PoS non ottiene buoni risultati, solo leggermente superiori alla baseline.

Risulta essere sottotono anche il modello che usa le informazioni linguistiche non lessicali, ottenendo solo il 60% di accuratezza, precision e recall e risultando il peggiore nella classificazione degli stereotipi.

Modello	Accuracy	Precision	Recall	F1-score
Informazioni linguistiche non lessicali	0.57	0.56	0.56	0.56
Bag of Words (forme)	0.65	0.65	0.65	0.65
Bag of Words (lemmi)	0.67	0.67	0.67	0.67
N-grammi di caratteri	**0.72**	**0.72**	**0.72**	**0.72**
N-grammi di parole	0.69	0.69	0.69	0.69
N-grammi di POS	0.59	0.59	0.59	0.59
Word Embeddings itWac (1 vettore)	0.67	0.67	0.67	0.67
Word Embeddings itWac (3 vettori)	0.64	0.64	0.63	0.64
Word Embeddings Twitter (1 vettore)	0.67	0.67	0.67	0.67
Word Embeddings Twitter (3 vettori)	0.65	0.65	0.64	0.64
Word Embeddings (Word2Vec)	0.62	0.62	0.61	0.60
Neural Language Model	0.71	0.71	**0.72**	0.71
Baseline	0.55	0.27	0.50	0.35

Tabella 27: Metriche di valutazione per i diversi modelli nella classificazione degli stereotipi

Nella classificazione degli stereotipi, il complesso modello BERT non produce il miglior risultato, che viene ottenuto dal più datato modello che usa gli n-grammi di caratteri.

Tutti i modelli hanno ottenuto risultati migliori rispetto alla baseline in entrambi i task. Va comunque sottolineato che la scelta del modello migliore dipende dal problema specifico e dalle esigenze dell'applicazione. In alcuni casi, potrebbe essere più importante avere una maggiore Precision; in altri casi, un valore più alto di Recall potrebbe essere più rilevante.

Riferimenti bibliografici

[1] Dominique Brunato et al. ≪Profiling-UD: a Tool for Linguistic Profiling of Texts≫. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. European Language Resources Association. Marseille, France, mag. 2020, pp. 7145–7151.

[2] Andrea Cimino, Lorenzo De Mattei e Felice Dell'Orletta. ≪Multi-task Learning in Deep Neural Networks at EVALITA 2018≫. In: *Proceedings of EVALITA '18, Evaluation of NLP and Speech Tools for Italian, 12-13 December, Turin, Italy*. 2018.

[3] Manuela Sanguinetti et al. ≪HaSpeeDe 2@ EVALITA2020: Overview of the EVALITA 2020 Hate Speech Detection Task≫. In: *Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020)*. A cura di Valerio Basile et al. Online: CEUR.org, 2020.