Toxic Wikipedia Comments Multilabel Classification Challenge

STUDENTS GROUP

Francesco Falleni, Francesco Gemignani

TEXT ANALYTICS PROJECT

Data Science & Business Informatics

Academic Year 2021/2022

Contents

1 Introduzione

La **Kaggle's challenge** affrontata in questo progetto riguarda il rilevamento e la classificazione di commenti con differenti tipologie di tossicità.

Blog, forum, la stessa **Wikipedia** dal quale questi commenti sono stati estratti e tutte le altre piattaforme che aggregano i contenuti degli utenti, sono l'origine della condivisione della conoscenza sul web. Tali informazioni sono molto utili per l'apprendimento e la consultazione, da parte non solo di studenti e insegnanti, ma di tutte quelle persone che necessitano di approfondire determinate tematiche. Per esempio, Wikipedia si basa sul contenuto generato dagli utenti e dipende dai feedback degli stessi per approvare i propri contenuti. Purtroppo però, molte persone non partecipano "civilmente" a queste discussioni, e colgono l'occasione per sfogare la loro rabbia e insicurezza, compromettendo sia la positività della **community** che la partecipazione alle discussioni.

L'**obiettivo** di questa challenge riguarda la rimozione (tempestiva) di questi commenti, al fine di mantenere un atteggiamento ottimista nelle discussioni dei vari topic.

Il dataset che ci è stato fornito classifica ogni commento con 6 differenti **categorie tossiche**, distinguendole in base al tipo di dannosità. Esse sono **toxic**, **severe toxic**, **threats**, **insults**, **obscene**, **identity hate**. Abbiamo definito due differenti task:

1. **Binary Toxicity Prediction - Balanced Learning**: nel quale alleniamo diverse tipologie di modelli, classici e non, al fine di classificare se un commento è appropriato o no. L'apprendimento è stato effettuato dopo aver bilanciato le classi.

2. **Multilabel Toxicity Prediction - Unbalanced & Balanced Learning**: task principale della competition che si pone come obiettivo quello di costruire dei modelli in grado di predire le categorie di tossicità di ogni commento (toxicity tagging). La label è composta da un vettore di 6 valori binari. L'apprendimento è stato eseguito sia sul training set originale che su quello bilanciato (rispetto alle classi di minoranza).

Il progetto è stato interamente sviluppato in **Python** e **Jupyter notebooks**. Abbiamo analizzato il dataset per capire la distribuzione delle labels e come esse sono correlate tra loro. E' stato eliminato il **rumore** da ogni commento (indirizzi emails, ip address, padding, ...) e le **stopwords**, riducento i termini rimanenti alla forma base (**stemming** e **lemmatization**). Utilizziamo sia approcci classici, come **regressione logistica**, **svm**, classificatori ensemble (**random forest** e **adaboost**), che modelli più avanzati come **Convolutional Neural Network**, **Recurrent Neural Network** e modelli pre-allenati come **BERT**.

Per maggiori dettagli e informazioni riguardo il dataset e la competition rimandiamo alla pagine della Kaggle's challenge (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge). Il dataset contiene un linguaggio estremamente offensivo che può emergere durante l'analisi.

2 EDA: Analisi dei Dati

In questo capitolo viene fatta l'analisi dell'intero dataset che ci è stato fornito, considerando tutti i commenti contenuti sia nel training che nel test set.

2.1 Generalità

Il dataset è contiene 312, 735 commenti e 8 features. Ogni record è composto da un identificatore, dalla stringa che contiene il testo del commento e da 6 labels binarie che rappresentano le diverse categorie di tossicità: toxic, severe_toxic, obscene, threat, insult e identity_hate. Abbiamo aggiunto una ulteriore feature **toxicity**, che indica se un commento è tossico o no, in particolare:

- assume valore 0 quando il commento non appartiene a nessuna delle 6 categorie di tossicità.
- altrimenti vale 1 quando il commento appartiene ad almeno una delle categorie.

Il dataset non contiene valori nulli ma da un'analisi generale abbiamo rilevato diverse labels settate a -1. Tali labels, contenute nel test set, identificano tutti quei commenti che non sono stati utilizzati nello scoring, quindi non sono stati presi in considerazione. Il dataset effettivo che analizziamo risulta composto da 223, 549 commenti e 9 features.

2.2 Distribuzione delle Labels

Sfortunatamente per il problema, ma fortunatamente per la Wikipedia community, i commenti tossici sono rari. Soltanto il 10.05% del dataset è etichettato come dannoso. Inoltre, alcune categorie tossiche sono estremamente rare e costituiscono meno dell'1% dei dati. I commenti non tossici coinvolgono invece l'89.95% di tutti i record.

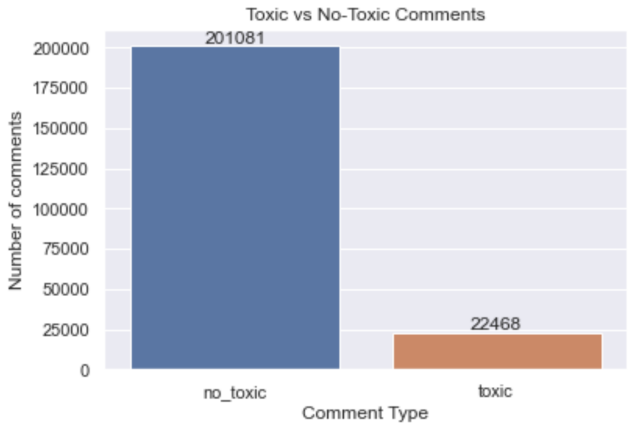


Figure 2.1: Frequenza delle label tossiche e non.

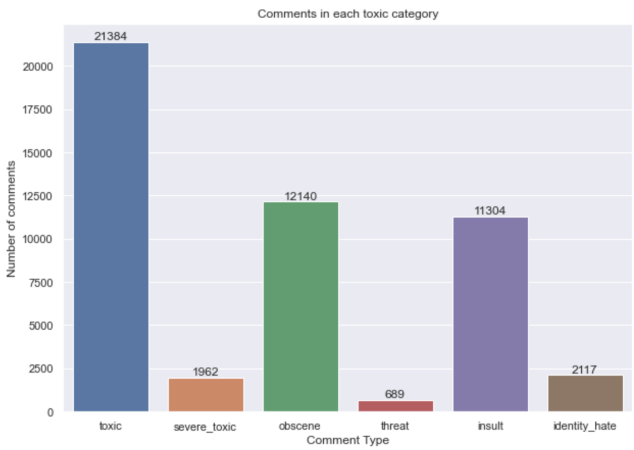


Figure 2.2: Distribuzione delle categorie tossiche.

Come possiamo vedere tra i commenti, le categorie che evidenziano lo sbilanciamento dei commenti tossici sono: *severe_toxic*, *threat* e *identity_hate*. Ogni commento tossico può essere

taggato con una o più categorie. Abbiamo rilevato che circa il 34% dei commenti tossici ha un'unica label, mentre la maggioranza dei commenti ha una sovrapposizione. Inoltre abbiamo notato che la classe toxic non cattura proprio tutti i commenti dannosi, infatti possono esistere alcuni commenti tossici (per esempio osceni o offensivi) che non sono taggati come toxic. Dalla correlation matrix possiamo vedere che è quasi improbabile che le minacce siano gravemente tossiche, né razziste mentre gli insulti tendono a essere spesso osceni.

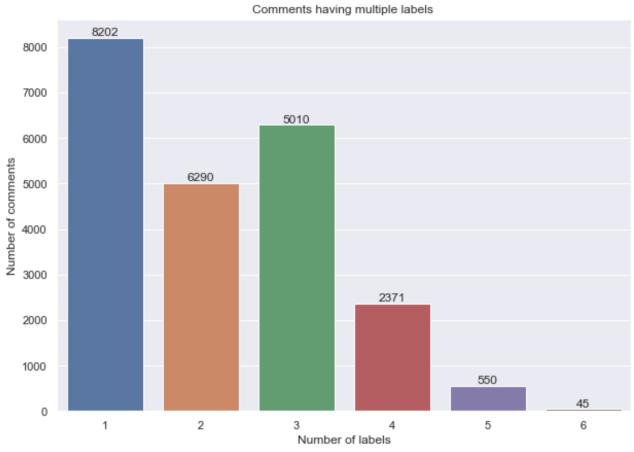


Figure 2.3: Multi-tagging dei commenti.

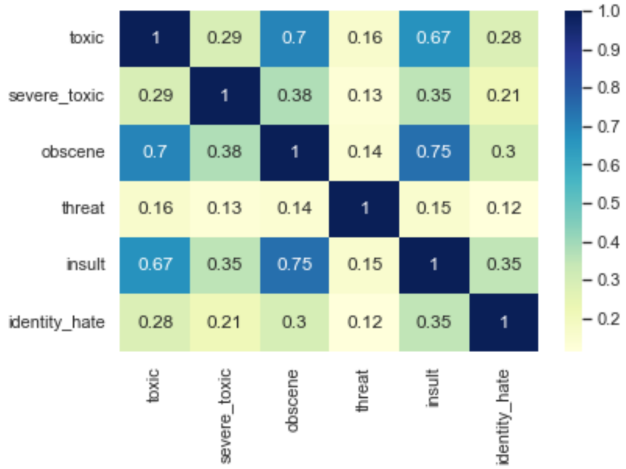


Figure 2.4: Correlazioni tra le labels tossiche.

2.3 Analisi dei Commenti

I commenti del dataset contengono molto rumore: sono composti da hashtags, emojies, urls, indirizzi email, indirizzi ip, etc. Ogni commento è composto in media da 391 caratteri e 66 termini. Abbiamo estratto i termini più rilevanti per ogni categoria di commento, notando che molti termini tossici vengono ripetuti in più categorie con una frequenza differente. Al termine dell'analisi abbiamo eseguito una fase di text cleaning eliminando tutto il rumore e successivamente costruito il vocabolario considerando più approcci (tokenizzazione con e senza stopword, lemmatization, stemming, bigrams, pos, etc). Questo ci ha permesso di allenare i modelli scegliendo la feature che meglio massimizza le performaces per un certo task.

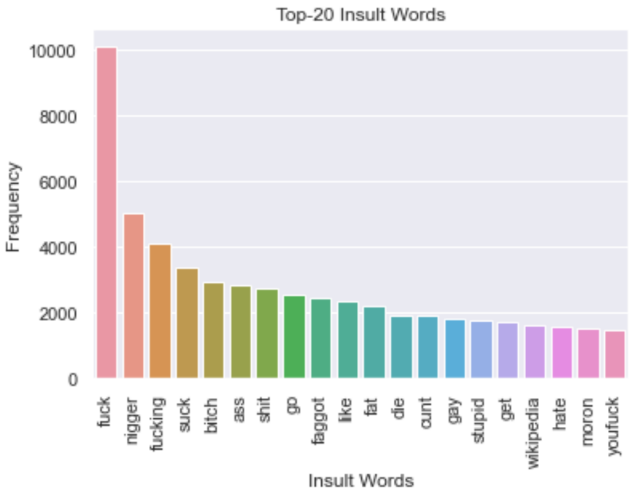


Figure 2.5: Top-20 insults.

Figure 2.6: Word Count plot sui commenti contenenti offese razziali e omofobiche.

3 Binary Toxicity Classification

Il task di classificazione binaria si pone come obiettivo quello di riuscire a predire se un commento è tossico o no. Un commento non è tossico se il vettore delle sei labels è nullo.

3.1 Preprocess dei Dati

Per addestrare i modelli abbiamo prima creato un unico dataset, contenente tutti i commenti del training e test set. Successivamente abbiamo eseguito l'undersampling così da bilanciare le labels tossiche e non, con un totale di 22,468 record per tipo. Abbiamo splittato i dati nel training set (70%) e test set (30%), stratificando le label così da preservarne le proporzioni.

Per quanto riguarda l'allenamento delle reti abbiamo aumentato le proporzioni del training set a 80% in quanto in fase di fitting, il 10% viene utilizzato come validation set. Il training set contiene $31,455$ commenti, quindi $15,728$ tossici e i restanti non tossici. Il test set è composto da $13,481$ record, di cui $6,741$ tossici e $6,740$ benigni.

Come accennato nel capitolo introduttivo, è stata generata una label sintetica **toxicity**, la quale assume valore nullo se il commento non è tossico, 1 altrimenti. In particolare la variabile tiene conto delle 6 categorie tossiche: se sono tutte uguali a 0 allora il commento è benigno, altrimenti se ne esiste solo una flaggata ad 1, tale commento viene giudicato tossico.

La metrica utilizzata per analizzare le performance dei modelli è l'accuracy, la quale fornisce un valore attendibile grazie al bilanciamento delle classi. Viene fatto riferimento anche alla AUC.

Infine, per decidere quale feature (tokenized_no_stopword, stemming, lemming, etc.) utilizzare per il training dei modelli abbiamo effettuato un test preliminare per capire quale approccio ritornava un accuracy maggiore, in particolare:

- nei modelli classici abbiamo usato lo stemming: i commenti sono stati privati del rumore, tokenizzati, privati di stopwords e ridotti con lo stemming. L'accuracy aveva un valore sensibilmente maggiori nello stemming rispetto alle altre variabili.
- nel training delle reti neurali abbiamo usato i commenti privati del rumore e tokenizzati con il metodo offerto dalla libreria Keras che trasforma ogni commento in una sequenza d' interi. In questo caso, si'e scelto 50.000 come numero massimo di parole da mantenere basato sulla frequenza. Infine, abbiamo effettuato un padding per assicurarci che ogni sequenza avesse la stessa lunghezza di 200

3.2 Modelli

Abbiamo eseguito il learning sia con modelli classici, che con approcci più avanzati. Di seguito descriviamo le linee guida per entrambe le classi.

3.2.1 Modelli Classici

Tra i modelli classici utilizzati per risolvere il task binario, abbiamo scelto il NaiveBayes classifier (MultinomialNB), Random Forest e Support Vector Machine (LinearSVC). Abbiamo eseguito

le trasformazioni in Pipeline, utilizzando CountVectorizer per costruire la matrice contenente il vocabolario di termini con le rispettive frequenze per ogni commento. Successivamente abbiamo trasformato la count matrix normalizzandola in base alla frequenza (tf) o all'inverse distance frequency (idf), usando TfidfTrasformer. Infine abbiamo passato lo space vector model a ogni classificatore senza eseguire il tuning dei parametri, così da avere un punto di riferimento dal quale migliorare il modello. Abbiamo quindi ottimizzato i parametri dei modelli testando 100 estimatori con GridSearch e RandomizedSearch in combinazione a 4 fold cross validation. Infine i modelli sono stati allenati nuovamente con i parametri ottimali e testati. Il motivo per cui abbiamo usato anche la RandomizedSearch è per ridurre i tempi più impegnativi causati dalla GridSearch. Tutti i modelli sono stati allenati sul training set e ottimizzati sullo stesso con 4-fold cross validation. Il test set è stato utilizzato infine per la valutazione delle performances.

Class	Precision	Recall	f1 score	support
No Toxic	0.88	0.87	0.87	6741
Toxic	0.87	0.88	0.88	6740

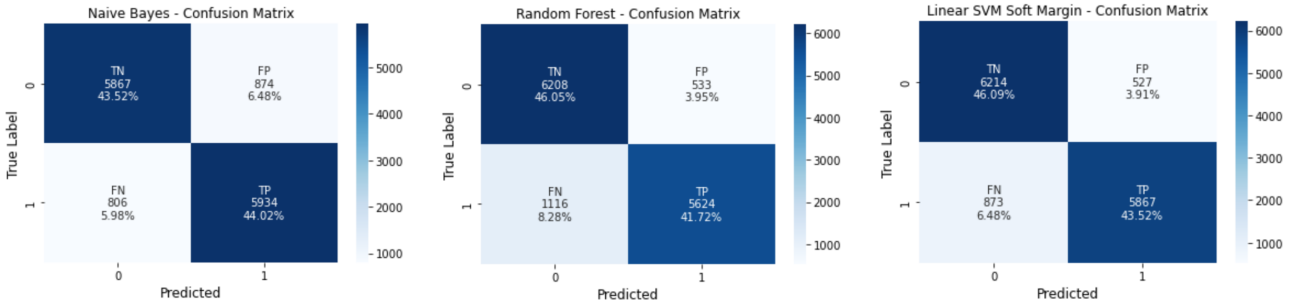
Table 3.1: Naive Bayes classification report con alpha=1. Accuracy=88%

Class	Precision	Recall	f1 score	support
No Toxic	0.85	0.92	0.88	6741
Toxic	0.91	0.83	0.87	6740

Table 3.2: Random Forest con criterion=gini e min_samples_split=5. Accuracy=88%

Class	Precision	Recall	f1 score	support
No Toxic	0.88	0.92	0.90	6741
Toxic	0.92	0.87	0.89	6740

Table 3.3: LinearSVC classification report con $C = 0.43$. Accuracy=90%



3.2.2 Recurrent & Convolutional Neural Networks

Per il training di entrambe le reti abbiamo utilizzato un layer di embedding in modo da trasformare il testo in una rappresentazione vettoriale più compatta per poter essere processata dalla rete. Come layer di embedding è stato utilizzato GloVe con 50 dimensioni, quindi ogni parola fornita in input alla rete viene rappresentata da un vettore con 50 valori.

Per le Recurrent Neural Network abbiamo utilizzato Long Short-Term Memory (LSTM) come unità ricorrenti in quanto sono una versione più evoluta delle tradizionali unità ricorrenti.

L'output del layer ricorrente, composto da 16 hidden units, è passato a un fully connected layer formato da 128 neuroni con funzione di attivazione *Relu*. Infine, il layer finale è composto da un solo neurone con funzione di attivazione *Sigmoid*.

Per quanto riguarda la Convolutional Neural Network abbiamo utilizzato dei convolutional layer intervallati da layer di max pooling, l'output poi passa ad un fully connected layer formato da 64 neuroni. Anche in questo caso il layer finale è composto da un solo neurone con funzione di attivazione Sigmoid.

Durante il training della rete abbiamo monitorato i valori di accuracy e loss ottenuti sul training set e sul validation set alla fine di ogni epoca (3.2.

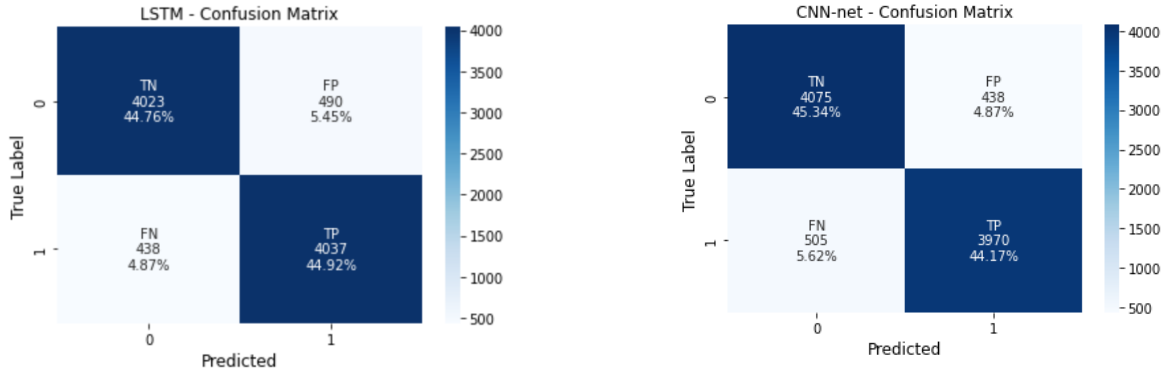
Per cercare di controllare l'overfitting delle reti abbiamo utilizzato dei layer di dropout e il meccanismo di early stopping monitorando il valore della loss sul validation set.

Class	Precision	Recall	f1 score	support
No Toxic	0.90	0.89	0.90	4513
Toxic	0.89	0.90	0.90	4475

Table 3.4: LSTM class report con hidden_size=16. Accuracy 90%.

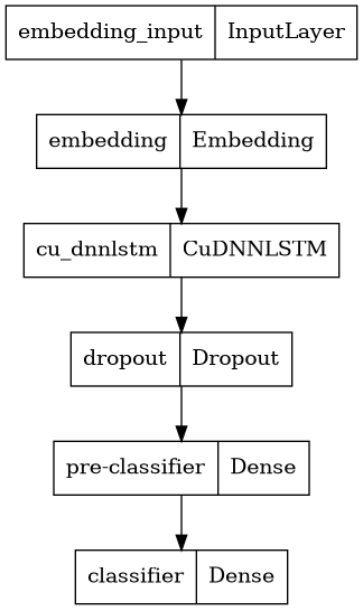
Class	Precision	Recall	f1 score	support
No Toxic	0.89	0.90	0.90	4513
Toxic	0.90	0.89	0.89	4475

Table 3.5: CNN class report con filters=16 e kernel_size=3. Accuracy 90%.

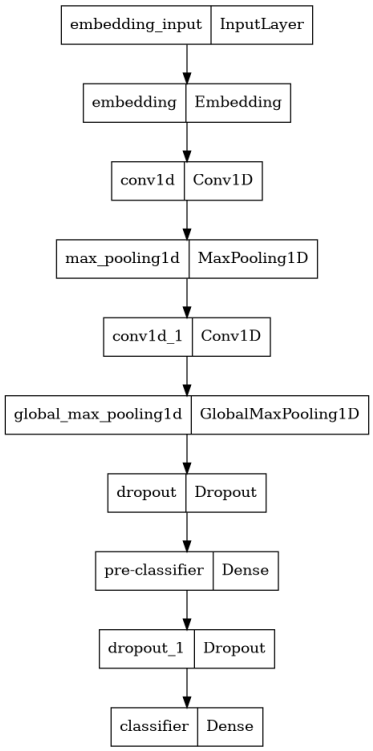


3.3 Valutazione delle Performances

Da un'analisi complessiva dei modelli allenati, i candidati con cui abbiamo ottenuto le performances migliori con SVM, CNN e LSTM, aventi tutti un'accuracy del 90% ed una stessa AUC. Le due reti hanno un bilanciamento migliore di precision e recall, in particolare LSTM ha una f1 score del 90% su entrambe le classi e valori di precision e recall leggermente migliori. A parità di performance, considerando la dimensione dello space vector model, potrebbe essere sensato scegliere il modello meno complesso come LinearSVC. In alternativa ricadere su LSTM.



(a) Architettura delle LSTM



(b) Architettura delle CNN

Figure 3.1

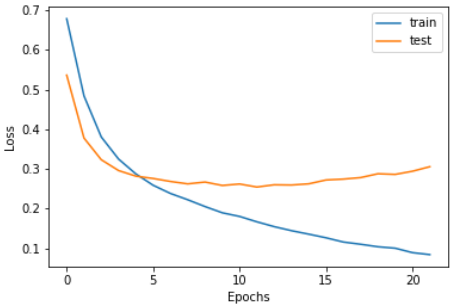


Figure 3.2: CNN loss sul training/validation set

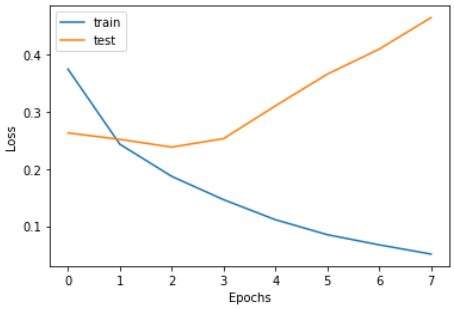


Figure 3.4: LSTM loss sul training/validation set

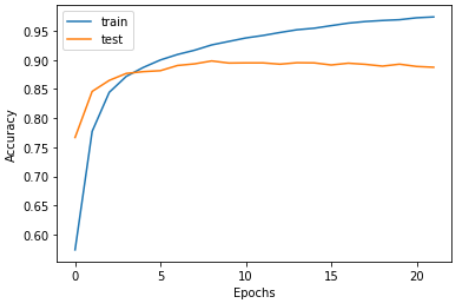


Figure 3.3: CNN accuratezza sul training/validation set

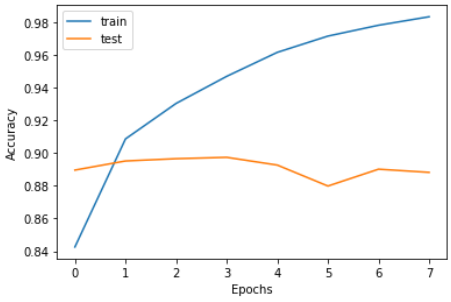


Figure 3.5: LSTM accuratezza sul training/validation set

4 Multilabel Toxicity Classification

Il task di classificazione multilabel consiste nel realizzare un modello in grado di riconoscere a quali categorie di tossicità un determinato commento appartiene. Per questo task abbiamo allenato algoritmi di machine learning classici e avanzati, considerando il training set originale e successivamente bilanciandolo rispetto alle labels minoritarie.

4.1 Data Preparation

A differenza del task binario, in questo caso abbiamo utilizzato il training/test set forniti da Kaggle per allenare tutti i modelli. Ricordiamo che il test set è stato privato di tutti i commenti non appartenenti allo scoring (toxic=-1). Il training set è composto da 159,571 commenti, di cui solo il 10% sono tossici, mentre il test set contiene 63,978 record, di cui tossici sempre il 10%. Entrambi i dataset risultano sbilanciati con una proporzione 1:10. Come possiamo vedere nella figura 2.1 abbiamo un forte sbilanciamento anche tra le categorie di commenti tossici: severe_toxic, threats e identity rate hanno una frequenza molto bassa.

Dato il forte sbilanciamento tra le labels, la metrica utilizzata per valutare le performance dei modelli è la f1 score.

Per decidere quale feature (stemming, lemming, etc.) utilizzare per il training dei modelli abbiamo effettuato un test preliminare dove valutiamo le prestazioni di un modello di logistic regression individualmente su ognuna delle feature generate. In particolare, abbiamo considerato il valore dello f1 score medio risultante da una cross validation a 5 fold. I risultati ottenuti sono mostrati in tabella 4.1.

	tokenized	tokenized no stop	stemming	PoS	lemming	lemming PoS	Bigrams
F1 avg	0.676	0.669	**0.678**	0.637	0.675	0.675	0.272
F1 micro	0.684	0.677	**0.687**	0.646	0.684	0.684	0.276
F1 macro	**0.502**	0.488	0.491	0.445	0.494	0.488	0.173

Table 4.1: f1 score di Logistic Regression con cross validation su ogni feature

Dai risultati possiamo individuare la feature *stemming* come la più promettente in termini di f1 score average e micro, questa feature è stata utilizzata per il training dei modelli di machine learning classici (SVM, LogisticsRegression, Adaboost).

Invece, per il training delle reti neurali abbiamo utilizzato il tokenizer predefinito di Keras, passando i parametri num_words = 75.000 e max_sequence_length=200.

4.2 Modelli

Abbiamo cercato di risolvere il problema utilizzando degli approcci classici di machine learning, poi passati a degli approcci basati su reti neurali ad hoc per il testo, e infine utilizzato un language model pre-allenato con fine-tuning.

4.2.1 Modelli Classici: Apprendimento Sbilanciato

Abbiamo utilizzato LogisticRegression, SVM e Adaboost per risolvere questo task. La strategia usata per affrontare il problema della classificazione multilabel è stata quella di fittare un classificatore per ognuna delle sei label che caratterizzano un commento. Ricordiamo che in questo caso la label è rappresentata da un vettore binario di 6 elementi. Tale scelta è stata realizzata utilizzando OneVsRest classifier. Come nel task binario, abbiamo effettuato tutte le trasformazioni in Pipeline, costruendo il vocabolario con CountVectorizer, trasformandolo con TfidfTransformer e passando la matrice a OneVsRest classifier, che a sua volta allena un classificatore per ogni label. Ogni modello utilizzato è stato allenato sul training set. Il tuning dei parametri lo abbiamo eseguito con RandomizedSearch in combinazione con una cross validation di 4 fold. Di seguito possiamo leggere i class report dei modelli riallenati con i parametri ottimizzati. Ricordiamo che la metrica utilizzata per valutare la bontà del classificatore è la f1 measure, in particolare avendo un dataset sbilanciato viene fatto riferimento alla f1 weighted avg che tiene conto anche del supporto di ogni classe.

Class	**Precision**	**Recall**	**f1 score**	**support**
toxic	0.58	0.79	0.67	6090
severe_toxic	0.39	0.39	0.39	367
obscene	0.69	0.68	0.68	3691
threat	0.38	0.27	0.32	211
insult	0.67	0.57	0.62	3427
identity_hate	0.59	0.34	0.43	712

Table 4.2: Logistic Regression classification report with C=11.2. F1_weighted_avg=0.64.

Class	**Precision**	**Recall**	**f1 score**	**support**
toxic	0.63	0.64	0.64	6090
severe_toxic	0.36	0.39	0.37	367
obscene	0.72	0.57	0.64	3691
threat	0.36	0.27	0.31	211
insult	0.68	0.45	0.54	3427
identity_hate	0.59	0.35	0.44	712

Table 4.3: Adaboost classification report con n_estimators=100. F1_weighted_avg=0.59.

Class	**Precision**	**Recall**	**f1 score**	**support**
toxic	0.58	0.80	0.67	6090
severe_toxic	0.38	0.35	0.36	367
obscene	0.67	0.69	0.68	3691
threat	0.37	0.29	0.33	211
insult	0.67	0.58	0.62	3427
identity_hate	0.62	0.36	0.46	712

Table 4.4: LinearSVC classification report con C=1.75. F1_weighted_avg=0.64.

4.2.2 Modelli Classici: Apprendimento Bilanciato

Nel tentativo d'incrementare le performance dei modelli abbiamo decrementato lo sbilanciamento delle label tossiche più frequenti rispetto alle minoritarie, eseguendo un oversampling

delle classi minoritarie: severe_toxic, obscene e identity_hate. Una volta bilanciato il dataset abbiamo riallenato gli stessi modelli utilizzati nel task sbilanciato appena descritti. Questo tentativo non ha portato a grandi cambiamenti. Come ci aspettavamo i valori della recall sulle classi minoritarie sono aumentati a discapito di un decremento della precision. In figura 4.5 sono riportate le valutazioni del classificatore con cui abbiamo riscontrato le performance migliori.

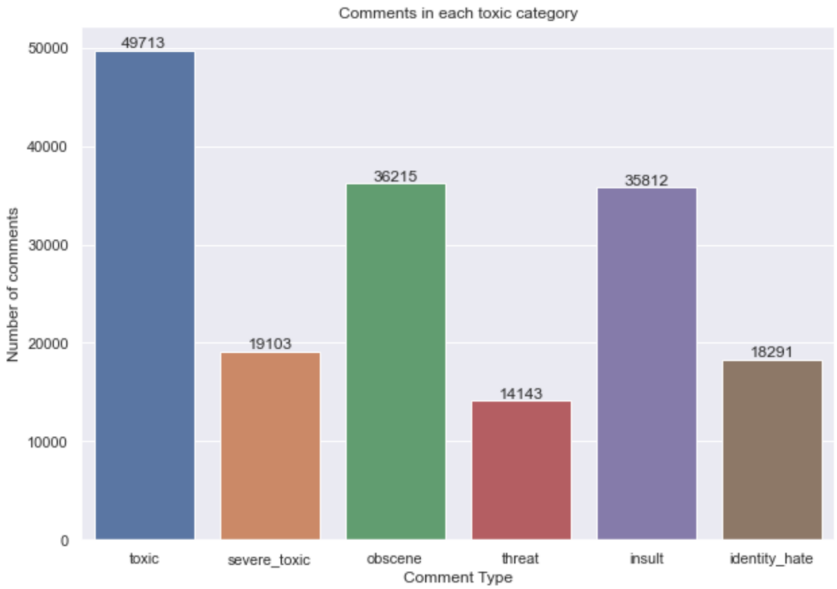


Figure 4.1: Label tossiche dopo l'oversampling delle classi minoritarie.

Class	Precision	Recall	f1 score	support
toxic	0.55	0.80	0.65	6090
severe_toxic	0.19	0.52	0.28	367
obscene	0.63	0.69	0.66	3691
threat	0.28	0.40	0.33	211
insult	0.57	0.56	0.56	3427
identity_hate	0.34	0.45	0.39	712

Table 4.5: LinearSVC classification report sul dataset bilanciato.
C=35.5, f1_weighted_avg=0.61, F1_micro_avg=0.60 e accuracy = 0.86

4.2.3 Modelli Avanzati: BERT, CNN & LSTM

Per il training delle reti abbiamo riservato il 20% del training set da utilizzare come validation set monitorando i valori di Precision, Recall e AUC ad ogni epoca.

Le architetture delle reti ricorrenti e convoluzionali sono analoghe a quelle descritte nel capitolo dedicato alla classificazione binaria 3.2.2. Le uniche differenze sono nell'utilizzo di alcuni parametri come hidden_size=32 per LSTM (prima era 16) e kernel_size=64 per CNN (prima era 16). Inoltre, abbiamo utilizzato un layer di embedding GloVe con 100 dimensioni (prima erano 50). Infine, Il layer finale è composto da 6 neuroni con funzione di attivazione *Sigmoid*, ognuno di essi rappresenta la probabilità di una categoria tossica.

Per quanto riguarda l'utilizzo di un language model pre-allenato abbiamo utilizzato due implementazioni lite di *BERT* fornite da TensorFlow:

1. L=2 hidden layers, a hidden size of H=768, and A=12 attention heads
2. L=4 hidden layers, a hidden size of H=768, and A=12 attention heads

L'architettura della rete basata su encoder BERT è strutturata nel seguente modo 4.2:

- *Input Layer*: layer di input per il testo
- *Tokenizer Layer*: layer di tokenizzazione del testo (pre-trained)
- *Encoder Layer*: bert encoder layer per trasformare il testo una rappresentazione vettoriale
- *Dropout Layer*: layer di dropout con probabilità 0.5
- *Pre-Classifier layer*: fully connected layer formato da 768 neuroni con funzione di attivazione *Relu*
- *Classifier Layer*: fully connected Layer formato da 6 neuroni con funzione di attivazione *Sigmoid*, ogni neurone corrisponde ad una categoria tossica.

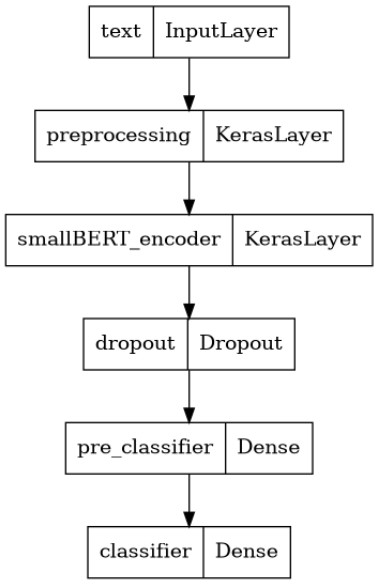


Figure 4.2: Bert fine tune architecture

Class	Precision	Recall	f1 score	support
toxic	0.50	0.85	0.63	6090
severe_toxic	0	0	0	367
obscene	0.75	0.60	0.67	3691
threat	0	0	0	211
insult	0.67	0.49	0.56	3427
identity_hate	0	0	0	712

Table 4.6: CNN classification report con filters=64 e kernel_size=5 F1_weighted_avg=0.57 and F1_micro_avg=0.60

Class	Precision	Recall	f1 score	support
toxic	0.45	0.89	0.59	6090
severe_toxic	0.37	0.30	0.33	367
obscene	0.59	0.73	0.65	3691
threat	0.34	0.07	0.12	211
insult	0.60	0.66	0.63	3427
identity_hate	0.52	0.45	0.49	712

Table 4.7: LSTM classification report con hidden_size=32 F1_weighted_avg=0.60 and F1_micro_avg=0.60

4.3 Valutazione delle Performances

È stata utilizzata la f1 measure per come metrica per la valutazione delle perfomances di ogni modello. In particolare, facciamo riferimento alla weighted avg, che tiene conto del support di ogni classe. Tra i modelli classici, pare ancora che SVM si comporti meglio; abbiamo una f1 weighted/micro pari al 65%. Quest'ultimo ha ottenuto risultati migliori anche rispetto al dataset con le labels bilanciate. Tra i modelli avanzati BERT si comporta decisamente meglio; rispetto a LinearSVC abbiamo un incremento del 2% della f1. Mentre i valori delle classi di maggioranza

Class	Precision	Recall	f1 score	support
toxic	0.53	0.89	0.67	6090
severe_toxic	0.33	0.59	0.42	367
obscene	0.58	0.82	0.68	3691
threat	0.44	0.64	0.52	211
insult	0.59	0.78	0.67	3427
identity_hate	0.56	0.71	0.62	712

Table 4.8: BERT classification report con L=2, H=768, A=12 F1_weighted_avg=66 and F1_micro_avg=66

Class	Precision	Recall	f1 score	support
toxic	0.52	0.91	0.66	6090
severe_toxic	0.36	0.56	0.44	367
obscene	0.59	0.81	0.69	3691
threat	0.46	0.52	0.49	211
insult	0.62	0.78	0.69	3427
identity_hate	0.62	0.64	0.63	712

Table 4.9: BERT classification report with L=4, H=768, A=12 F1_weighted_avg=67 and F1_micro_avg=67

sono pressoché simili ad SVM, le classi di minoranza vengono classificate decisamente meglio. Rispetto al modello classico, la rete pre-allenata con 4 hidden layer ha incrementato la f1 score della classe threat del 22%, del 18% la classe identity hate e del 9% severe toxic. I miglioramenti si rispecchiano anche nella precision e recall. Sarebbe stato interessante valutare il numero di hidden layer ottimale al fine di massimizzare le performances senza andare in overfitting, di conseguenza distribuire l'apprendimento su più macchine. L'architettura dei calcolatori sul quale abbiamo allenato le reti non ci ha permesso di incrementare tali parametri, con cui avremmo potuto ottenere una capacità di generalizzazione migliore.

5 Conclusioni

La Kaggle challenge che abbiamo affrontato in questo progetto si pone come obiettivo quello di rilevare e classificare commenti con diversi tipologie di tossicità. Tali commenti sono stati estratti da Wikipedia, la quale si basa sul feedback degli stessi per approvare i propri contenuti. Di conseguenza, al fine di mantenere un atteggiamento positivo nelle discussioni dei vari topic, è necessario individuare e rimuovere in modo efficiente i commenti più dannosi. Oltre al task principale della competition, ne abbiamo aggiunto uno ulteriore. Ci siamo posti il problema di identificare semplicemente se un commento è tossico o no, indipendentemente dalle tagging toxicity labels con cui viene identificato. A tal proposito abbiamo realizzato diversi modelli di machine learning, classici e più avanzati. I risultati ottenuti dopo aver bilanciato il dataset sono stati complessivamente positivi, con poche differenze tra un modello e l'altro, restituendo una accuratezza tra l'88-90%. Differente la situazione nel caso multilabel toxicity prediction. Anche in questo caso abbiamo implementato differenti modelli al fine di predire il vettore delle label. Non soddisfatti dei risultati, abbiamo provato a bilanciare le classi tossiche minoritarie per capire se ne potessero derivare dei miglioramenti. I risultati ottenuti, anche in questo caso, non sono stati migliori rispetto al task bilanciato. Questo potrebbe far escludere il problema dello sbilanciamento ma non possiamo affermarlo con certezza. In generale, abbiamo ottenuto le performance migliori con i modelli pre-allenati come BERT, con una f1 score poco inferiore al 70%. Un interessante tentativo sarebbe quello di provare ad utilizzare più layer nei modelli pre-allenati, quindi distribuire i calcoli su più macchine.