Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Classification of Toxic Online Comments

Authors:
Luca Gandolfi - 807485 - l.gandolfi3@campus.unimib.it
Stefano Sacco - 807532 - s.sacco3@campus.unimib.it

January 16, 2020

Abstract

L'analisi e l'individuazione automatica di contenuti testuali negativi online è un tema caldo degli ultimi anni nel Natural Language Processing. Questo progetto ha lo scopo di costruire un modello di apprendimento basato su una Deep Neural Network in grado di classificare commenti online secondo sei differenti tipologie di tossicità. Lo studio del modello ha permesso di utilizzare diverse rappresentazioni del testo, come Word Embedding pre-trained e feature ottenute tramite Transfer Learning applicato ad uno dei migliori modelli allo stato dell'arte nell'ambito della Text Classification. I risultati ottenuti sono stati confrontati e analizzati, mostrando i problemi emersi.

1 Introduzione

Nel corso degli ultimi anni, il Natural Language Processing ha subito una crescita importante, sviluppando tecniche sempre più efficienti per processare e classificare testi espressi in linguaggio informale. Restano, tuttavia, ancora aperti alcuni temi caldi che richiedono uno studio più accurato, come ad esempio l'individuazione dei così detti commenti *tossici*, ovvero espressioni offensive che richiedono una censura o addirittura di essere rimosse.

Con l'esplosione della popolarità dei Social Network, chiunque ha libertà di esprimere il proprio pensiero su qualsiasi piattaforma. Il problema non riguarda l'utilizzo di singole "parolacce", censurabili tramite una black list di parole, ma piuttosto espressioni più complesse, come frasi con sottointeso odio razziale, minacce, satira offensiva, etc.

Lo studio presentato con questo progetto si pone il compito di sperimentare tecniche di analisi testuali tramite modelli di Machine Learning e testare quali migliorie sono state introdotte con i recenti studi allo stato dell'arte, come l'ormai popolare Google BERT. Nel dettaglio, verranno utilizzate tecniche di pre-processing prima di allenare una Deep Neural Network costruita esclusivamente con il fine di individuare diverse tipologie di tossicità nei commenti online.

2 Dataset

Il dataset utilizzato per questo progetto è **Toxic Online Comments** [1]. Il dataset, suddiviso tra *train set* e *test set*, è composto da commenti effettuati su Wikipedia Talk pages [2], etichettati in base a diversi tipi di tossicità. Un commento è tossico quando presenta espressioni offensive, o più in generale negative, nei confronti di qualcuno o qualcosa. Nel dettaglio, il dataset è descritto da 2 attributi e 6 etichette. Gli attributi sono un identificativo del commento (superfluo al fine di classificazione) e il testo del commento. Le 6 classi sono tutte binarie e indicano la tipologia di tossicità del commento. I commenti non tossici presentano tutte le classi con valore zero.

I tipi di tossicità presenti sono:

- **Toxic**: molto cattivo, sgradevole o dannoso.
- **Severe toxic**: estremamente cattivo e offensivo.
- **Obscene**: offensivo o disgustoso secondo gli standard di moralità e decenza accettati.
- **Threat**: una dichiarazione di intenzione di infliggere dolore, ferire, danneggiare o altree azioni ostili su qualcuno come castigo per qualcosa fatto o non fatto.
- **Insult**: parlare o trattare con abuso irriverente o sprezzante.
- **Identity hate**: odio, ostilità o violenza nei confronti di membri di una razza, etnia, nazione, religione, genere, identità di genere, orientamento sessuale o qualsiasi altro gruppo della società.

Un commento può essere descritto da più etichette simultaneamente, indicando che le espressioni utilizzate possono rientrare in diverse categorie di tossicità.

2.1 Analisi preliminari e creazione del Test set

Il Train set è estremamente sbilanciato verso commenti non tossici. Ovviamente, una tale situazione è un forte svantaggio per il modello di apprendimento. In particolare, circa l'89% del dataset è composto da commenti non tossici, questa forte differenza è osservabile in Figura 1. Il Train set verrà successivamente suddiviso tra Train e Validation set prima della fase di training. La colonna relativa all'identificatore dei commenti è stata, invece, rimossa subito, ottenendo quindi un dataset contenente soltanto 1 attributo (il testo del commento) e 6 etichette.

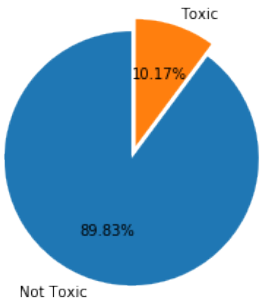


Figura 1: Confronto proporzioni tra commenti tossici e non tossici.

Come è possibile osservare in Tabella 1, i commenti totali sono 159 571. Considerando che un tale numero permetta alla Deep Neural Network di apprendere senza problemi la classificazione di un commento normale (i.e. label tutte a zero), dal Test set sono stati estratti tutti i commenti etichettati come tossici ed è quindi stato

Tabella 1: Distribuzione del Train set.

Toxic comments	16 225
Not Toxic comments	143 346
Total comments	159 571

costruito un nuovo Test set composto esclusivamente da questi. Nel dettaglio, il nuovo Test set è composto da 6 243 commenti.
Approfondendo l'analisi dei commenti tossici nel Train set, è possibile osservare il numero di esempi per ogni tipologia. Questa prima analisi ignora il fatto che un commento possa essere descritto da più etichette, basando il calcolo soltanto sulla presenza di un 1 nella corrispondente classe. E' possibile osservare graficamente le proporzioni tra le classi in Figura 2, e numericamente in Tabella 2.

Tabella 2: Quantità di commenti per ogni tipologia di tossicità.

Tipologia di Tossicità	**Commenti**
Toxic	21 384
Severe Toxic	1 962
Obscene	12 140
Threat	689
Insult	11 304
Identity Hate	2 117

Dai risultati emerge come la classe Toxic sia la più popolata. Anche le classi Obscene e Insult mostrano numeri importanti. Bisogna tener presente però, che la classe Toxic è presente anche in molti commenti caratterizzati da più etichette, quindi la proporzione è in parte giustificata. Una possibile considerazione veloce riguarda il numero estremamente ridotto di commenti etichettati come Threat: è difficile trovare minacce su un dominio come Wikipedia Talk. Proporzioni simili sono state riscontrate anche nel Test set.

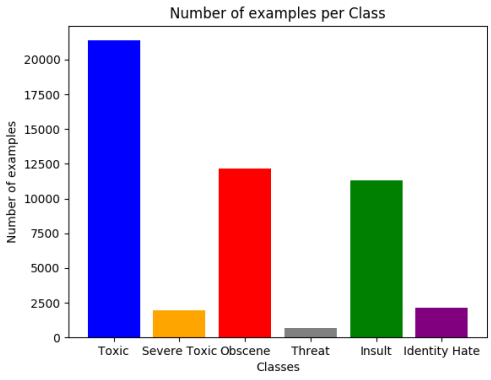


Figura 2: Distribuzione commenti per tipologia di tossicità.

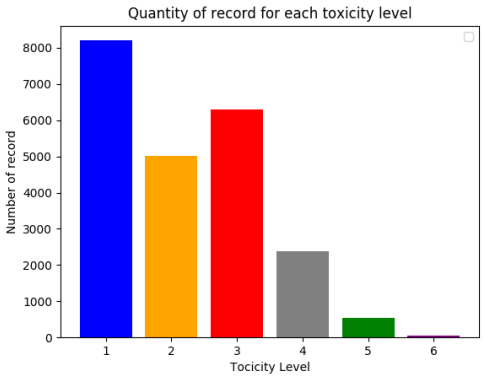


Figura 3: Distribuzione commenti per numero di etichette.

Considerando la presenza di più etichette descrittive in molti commenti tossici, è stato scelto di approfondire ulteriormente questo aspetto, andando a valutare il numero di commenti che presentano una o più etichette. Vi sono commenti descritti addirittura da tutte e 6 le etichette. Anche per questa analisi è possibile osservare graficamente le proporzioni in Figura 3, e numericamente in Tabella 3.

Tabella 3: Quantità di commenti per numero cumulativo di etichette.

Numero Etichette	**Commenti**
1 etichetta	8 202
2 etichette	5 010
3 etichette	6 290
4 etichette	2 371
5 etichette	550
6 etichette	45

Come è possibile osservare, la maggior parte dei commenti è caratterizzata da una sola etichetta. Tuttavia, sono presenti molti commenti con due, tre e quattro etichette. Commenti con 5 e 6 etichette sono invece una minoranza. Questa vasta distribuzione potrebbe portare a difficoltà in fase di training e sarà una importante caratteristica da tenere in considerazione.

3 Approccio Metodologico

L'approccio metodologico seguito per affrontare il progetto può essere diviso in 5 step: (1) Text Preprocessing con il fine di pulire il testo dei commenti da tutte le informazioni superflue, (2) WordEmbedding per ottenere una rappresentazione vettoriale e migliorare le capacità di apprendimento, (3) creazione e addestramento di una Deep Neural Network, (4) Data Augmentation sulla classe di minoranza per provare a migliorare i risultati ottenuti con la rete, (5) Transfer Learning utilizzando Google BERT, con relativa fase di ottimizzazione. Ognuno di questi step verrà analizzato nel dettaglio nelle seguenti sottosezioni.

3.1 Text preprocessing

Una collezione di testi in linguaggio informale è, in genere, un limitatore di performance per modelli di classificazione e di analisi. Un task fondamentale nel Natural Language Processing (NLP) è il Preprocessing dei testi, con il fine di rendere i dati più chiari per una macchina. La fase di Preprocessing può variare molto a seconda del task che si vuole svolgere. In questo progetto, come già anticipato, è necessario preparare i testi per essere classificati in 6 classi di tossicità. E' necessario, dunque, individuare dei descrittori all'interno dei testi, in grado di discriminare le classi. La scelta di utilizzare un modello pre-trained di WordEmbedding ha inoltre influito sulle scelte prese e sulle funzioni utilizzate.

Come prima operazione, i testi dei commenti sono stati portati in *lower case* e successivamente sono state rimosse le contrazioni della lingua inglese (ad esempio

la parola *i'm* è stata trasformata in *i am*), al fine di ottenere un testo più pulito. A questo punto, i testi sono stati alleggeriti dalle *Stop words*, ovvero parole usate per collegare parti di testo come ad esempio articoli, congiunzioni, persone prima dei verbi, etc, le quali sono, in genere, le parole più frequenti e quindi cattivi descrittori. La parola *not* non è stata rimossa. Il motivo dietro questa scelta riguarda la considerazione che spesso è una parola fondamentale nella Sentiment Analysis per quanto riguarda il sentiment negativo, ed è possibile che sia importante anche per individuare espressioni offensive complesse, non limitate soltanto a insulti, come ad esempio minacce. L'insieme delle stopwords, preso da NLTK, è stato arricchito con ulteriori imprecisioni trovate nei testi, ad esempio la conversione in testo del carattere "a capo". Infine, è stata effettuata la *Tokenization* del testo, ovvero il testo è stato suddiviso in singoli token, dove ogni token rappresenta una parola (unigram). La libreria utilizzata ha permesso di evitare di rimuovere manualmente la punteggiatura, fornendo dei filtri di default per svolgere direttamente questo task.

3.2 WordEmbedding

Preventivamente alla fase di WordEmbedding, è stato applicato un *post Padding* ai testi dei commenti Tokenizzati. Questo ha permesso di ottenere una lunghezza univoca pari a 200 per tutti i commenti inserendo degli 0 in coda, nel caso di lunghezza inferiore. Commenti tokenizzati di lunghezza maggiore a 200 sono stati troncati alla grandezza univoca. Per la creazione di WordEmbedding è stato utilizzato **GloVe** [3]. In particolare, è stato scelto di utilizzare i pesi appresi da un modello allenato su un dataset di 400mila vocaboli, per massimizzare le performance (i vocaboli nel dataset utilizzato in questo progetto non superavano i 230mila). I pesi di GloVe sono stati utilizzati direttamente nella Deep Neural Network, costruendo un layer di Embedding che riceve in input i commenti tokenizzati e restituisce in output una loro rappresentazione vettoriale in 200 dimensioni per ogni token. La scelta di utilizzare vettori di 200 dimensioni è stata presa per permettere di ottenere maggiore espressione possibile senza richiedere troppo tempo alla rete per processare dimensioni maggiori. In Figura 4 viene riassunta la fase di preprocessing, includendo la creazione di vettori tramite Word Embeddings.

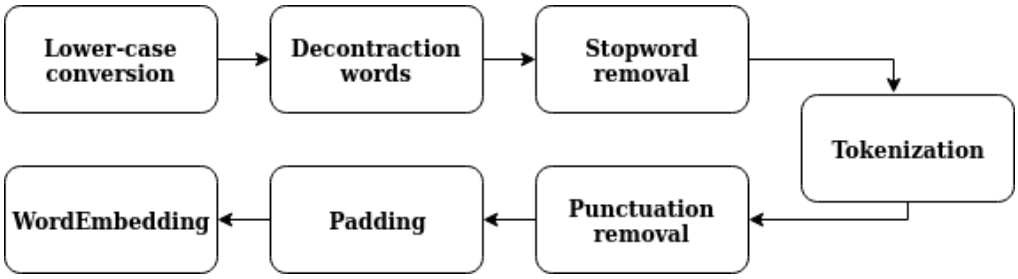


Figura 4: Fase di Text-Preprocessing nel dettaglio.

3.3 Deep Neural Network

La Rete Neurale è stata costruita utilizzando **TensorFlow** e, in particolare, **Keras**. Come già anticipato, il layer di input utilizzato è stato il layer di Embedding con il vincolo sul non allenare i propri pesi, forniti dal sito web di GloVe. L'output di tale layer ha richiesto una operazione di *Flattening* per poter utilizzare i vettori in WordEmbedding nei successivi layer densi fully-connected. Il design della rete creata manualmente è composto da 2 Hidden Layer più un Output Layer. L'ottimizzatore utilizzato è Adam, in quanto è uno dei più usati per svolgere Text Classification. Adam è stato impostato con l'opzione amsgrad [4] attiva e con un Learning Rate custom. Nel dettaglio, è stato inserito un *monitor* sulla funzione di Validation Loss, al fine di impostare un *Learning Rate* adattivo. Il funzionamento del monitor è semplice: controlla il valore della Validation Loss e, se non ottiene un miglioramento dopo 2 epoche, decrementa il Learning Rate in modo tale da permettere alla funzione di addentrarsi in una zona di minimi locali, al fine di individuare il minimo globale.

Per quanto riguarda il primo *Hidden Layer* dopo quello di Embedding, sono stati utilizzati 100 neuroni con funzione di attivazione ReLU e, per il secondo *Hidden Layer*, 50 neuroni con la stessa funzione di attivazione. Infine, è stato definito un Output Layer con soltanto 6 neuroni: uno per ogni etichetta. In questo caso, è stata utilizzata una funzione di attivazione Sigmoidale, per scalare l'output in range di probabilità (tra zero e uno). In coppia alla Sigmoid, scelta quasi obbligata l'utilizzo di una Binary Crossentropy come Loss Function. In Figura 5, è possibile osservare uno schema ad alto livello raffigurante i layer utilizzati nel modello.

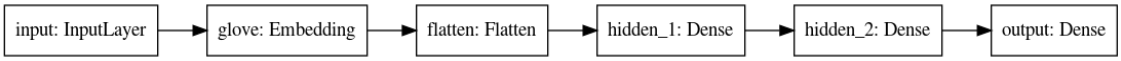


Figura 5: Layer utilizzati nel modello costruito.

Il modello è stato appreso per 18 epoche, utilizzando una dimensione di batch pari a 256, un numero in realtà piccolo rispetto alla dimensione del dataset ma che ha permesso di ottenere i risultati migliori. Tramite l'utilizzo del Model Checkpoint, è stato possibile monitorare i pesi appresi durante le diverse epoche, andando a salvare la migliore configurazione in termini di Validation F1-measure, in quanto si voleva massimizzare il rapporto tra Precision e Recall per le sei classi. In Tabella 4, è possibile osservare un sommario del modello, descrittivo del numero di parametri e della forma dell'output di ogni layer. La presenza di un layer di Embedding ha aumentato in modo importante il numero complessivo di parametri del modello.

3.4 Data augmentation

La scarsa quantità di alcune tipologie di commenti tossici ha portato a considerare tecniche di Data Augmentation, con il fine di migliorare i risultati ottenuti dalla rete neurale descritta nella sezione 3.3. Per svolgere questo task, è stata creata una matrice utilizzando un approccio Nearest Neighbors a partire dalla matrice di Embedding ottenuta tramite GloVe. Questa matrice è stata utilizzata per creare un

Tabella 4: Summary del modello.

Layer	# Parametri	Output shape
Input	0	(, 200)
GloVe Embedding	51 464 200	(, 200, 200)
Flatten	0	(, 40 000)
Dense 1	4 000 100	(, 100)
Dense 2	5 050	(, 50)
Output	306	(, 6)
Totale	55 469 656	
Allenabili	4 005 456	

dizionario di *sinonimi* delle parole presenti nel dataset. Attraverso il dizionario dei sinonimi, sono stati creati dei commenti artificiali a partire da commenti etichettati come severe toxic e threat, poichè caratterizzati da scarso supporto. Per ognuna delle frasi sono state generate due nuove frasi artificiali. Al termine di questa fase i 3922 nuovi commenti generati, sono stati, quindi, aggiunti al train set ed etichettati nel modo corretto, ottenendo così un nuovo dataset composto da 163 493 esempi. Infine, è stata utilizzata la rete neurale precedentemente descritta per apprendere il modello sul nuovo dataset.

3.5 Transfer Learning

Il Transfer Learning è una tecnica di Machine Learning che permette di utilizzare la conoscenza acquisita da un precedente task come punto di partenza per affrontarne uno nuovo. Il nuovo task può essere più o meno correlato al precedente. L'idea di base è, quindi, quella di sfruttare una Rete Neurale precedentemente allenata e utilizzare i pesi appresi, con il conseguente beneficio di non dover creare un nuovo modello da zero. In questo progetto è stato utilizzato il Transfer Learning in ottica *Feature Extraction*, andando ad utilizzare il modello scelto senza dover far apprendere i pesi, semplicemente prendendone l'output. Come modello pre-trained è stato utilizzato Google **BERT** [5]. BERT (Bidirectional Encoder Representations from Transformers), è un modello transformer-based bidirezionale pre-allenato, sviluppato utilizzando una combinazione di due task denominati Masked Language Modeling e Next Sentence Prediction su un corupus di elevate dimensioni, in maniera *unsupervised*. Allo stato dell'arte è uno dei migliori modelli utilizzabili su svariati task di NLP.

Per utilizzare BERT, è stato necessario effettuare un pre-processing ai commenti diverso rispetto a quello trattato nei paragrafi 3.1 e 3.2. In particolare, i testi dei commenti sono stati soltanto convertiti in lower-case e alleggeriti dalle contrazioni della lingua inglese. Per ottenere le rappresentazioni codificate, ovvero Sentence Encoding, è stata utilizzata una soluzione di Bert-as-Service proposta in [6], la quale prevede di settare pochi parametri, come il numero massimo di token da considerare per ogni commento: 512. Come modello pre-trained è stato utilizzato BERT-Base, che fornisce in output vettori di 768 dimensioni.

E' stato inizialmente considerato l'utilizzo di BERT in ottica Fine-Tuning, tuttavia per limitazioni hardware è stato deciso di estrarre delle codifiche dei commenti ed osservare se tramite queste codifiche, fosse possibile migliorare la classificazione. L'estrazione dei vettori ha richiesto un tempo di circa 48 ore.

Per classificare le codifiche dei commenti, è stata utilizzata una rete leggermente diversa da quella descritta nella sezione 3.3, con 3 hidden layer più un output layer e un Dropout per evitare overfitting. Iperparametri come il numero di neuroni, la dimensione di batch e il numero di epochs sono stati oggetto di Ottimizzazione e verranno discussi nella prossima sezione. Anche in questo caso è stato utilizzato un monitor sul Learning Rate.

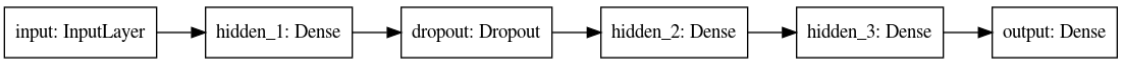


Figura 6: Layer utilizzati nel modello per BERT.

In Figura 6, è possibile osservare il modello definito ad alto livello, mentre in Tabella 5, è possibile analizzare le dimensioni dei neuroni in seguito alla fase di ottimizzazione. Altri iperparametri come Activation function, Loss function e Optimizer non hanno subito cambiamenti rispetto alla prima rete.

3.6 Hyper-parameters Optimization

Considerando l'evidente numero ridotto di parametri nella rete che prende in input le rappresentazioni ottenute tramite Transfer Learning (768 feature in input contro le 40 mila delle altre reti soltanto nel primo layer denso) è stato scelto di effettuare un processo di ottimizzazione degli iperparametri (HPO) soltanto su questa rete. La libreria utilizzata è Talos [7], che permette analisi indirizzate allo scopo di ridurre lo spazio di ricerca degli iperparametri.

Il problema di classificazione multi-label impedisce di utilizzare una strategia efficiente di Cross Validation Stratificata. L'utilizzo di una normale CV, invece, in una situazione di sbilanciamento di classi come questa, comporterebbe il rischio di avere dei fold senza elementi di una classe di minoranza, ad esempio Threat, andando a complicare l'analisi. Per queste ragioni, dopo alcuni test, è stato deciso di ottimizzare senza una Cross Validation, ma di utilizzare una buona suddivisione tra Train e Validation attraverso un seed.

Il task di HPO è stato svolto in due fasi. Entrambe le fasi hanno visto l'utilizzo di Random Forest come metodo di reducer, e la Validation Fmeasure come metrica da ottimizzare. Gli iperparametri in esame sono stati 5: numero di epochs, numero di neuroni nei 3 hidden layer, dimensione di batch.

3.6.1 Fase 1: Analisi degli iperparametri

La prima fase è stata caratterizzata da un budget fortemente limitato, in particolare a soltanto 4 iterazioni, con il fine di avere un'idea generale su quali iperparametri influiscano maggiormente sulla metrica in esame, ovvero Validation Fmeasure (nonostante sia indifferente in questo caso la Fmeasure del train).

Come è possibile osservare in Figura 7, i risultati di questa prima esecuzione hanno evidenziato, per la Val Fmeasure, una correlazione diretta con la dimensione di batch e il numero di epochs, mentre una correlazione inversa con il numero di neuroni nei vari layer e con il learning rate. Tralasciando questo ultimo iperparametro, che è adattivo in base alla Val Loss, è stato deciso di ridurre lo spazio degli altri iperparametri rimuovendo basse dimensioni di batch e bassi numeri di epochs, poichè al crescere di queste ultime è stata notata una crescita della metrica in esame. Discorso inverso invece per il numero di neuroni nei 3 hidden layer.

Ulteriori analisi svolte, come lo studio del Kernel Density Estimation, sono visibili sul terzo notebook consegnato.

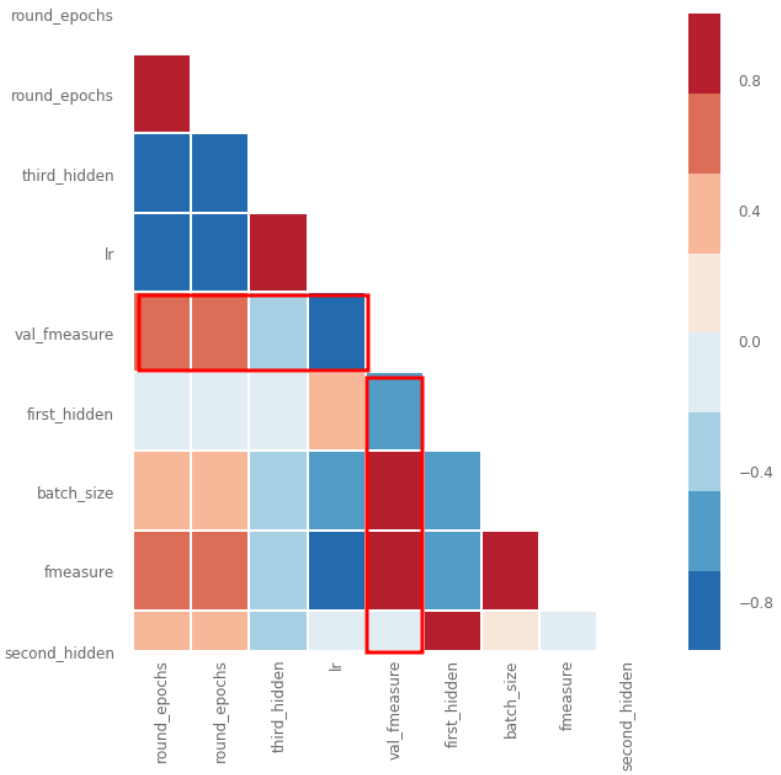


Figura 7: Analisi di correlazione tra Val Fmeasure e gli iperparametri.

3.6.2 Fase 2: Individuazione della combinazione migliore

La seconda fase è stata caratterizzata da una nuova esecuzione con un budget superiore, in particolare 16 esecuzioni, con lo spazio degli iperparametri ridotto dopo la Fase 1. La seguente configurazione si è rivelata essere la migliore: 15 epochs, 256 di batch size, 500, 200 e 50 neuroni rispettivamente per il primo, il secondo e il terzo hidden layer. In Tabella 5 è possibile osservare il numero di parametri del modello ottimizzato.

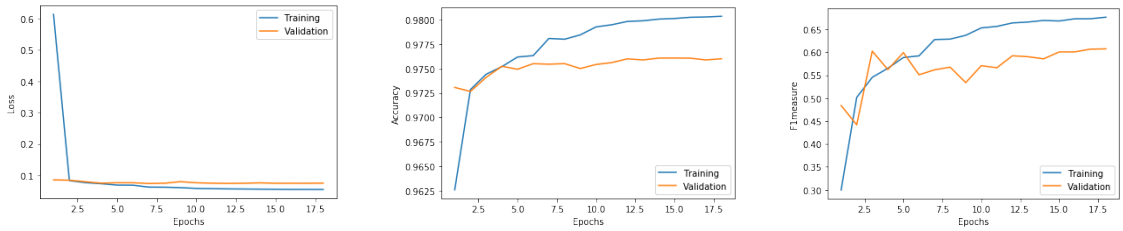
Tabella 5: Summary del modello ottimizzato.

Layer	# Parametri	Output shape
Input	0	(, 768)
Dense 1	384 500	(, 500)
Dropout	0	(, 500)
Dense 2	100 200	(, 200)
Dense 3	10 050	(, 50)
Output	306	(, 6)
Totale	495 056	

4 Risultati e valutazioni

In questa sezione vengono mostrati i risultati relativi ai 3 esperimenti svolti durante il progetto, mentre la loro analisi verrà effettuata nel capitolo 5. E' bene anticipare che la metrica target utilizzata nei vari esperimenti è la F1-score micro average, poichè il dataset non solo è sbilanciato fortemente verso commenti non tossici, ma presenta ulteriori sbilanciamenti tra le varie classi di tossicità, in particolare le classi Severe Toxic e Threat sono scarsamente popolate. I risultati verranno mostrati anche in termini di Accuracy e di Loss, per dimostrare come il vero problema risieda nel riconoscimento delle 6 diverse categorie di tossicità dei commenti, più che nella corretta classificazione di commenti non tossici. Inoltre, come descritto nella sezione 2.1, il Test set utilizzato è composto da soli commenti tossici, poichè lo scopo finale del progetto è la definizione di un classificatore in grado di analizzare i testi ed assegnarli a 6 diverse tipologie di tossicità.

Il primo esperimento è stato caratterizzato dall'apprendimento di un semplice modello definito con la rete descritta nella sezione 3.3, preceduto da un task di Pre-Processing e di Word Embedding per generare i vettori numerici. Come è possibile osservare in Figura 8, Accuracy e Loss presentano valori ottimali. Il problema di questo modello però, risulta essere il rapporto tra Precision e Recall, mostrato in Tabella 6.



(a) Loss. **(b)** Accuracy. **(c)** F1-score.

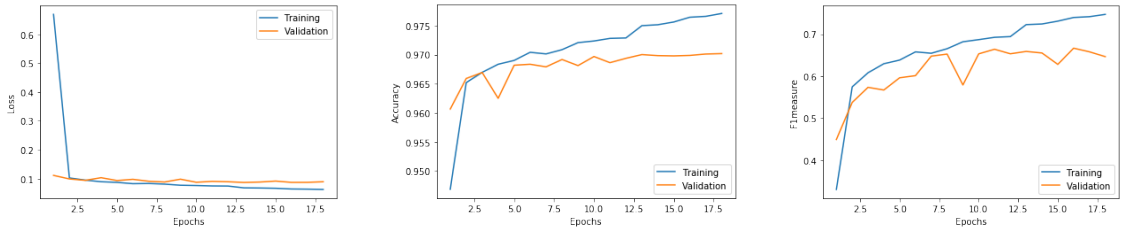
Figura 8: Andamento delle metriche principali durante il training della prima rete.

I pessimi risultati ottenuti con questa prima rete, hanno portato alla definizione di due possibili soluzioni: (1) lavorare sul dataset applicando oversampling ad alcune

Tabella 6: Precision, Recall e F1-score per la prima rete definita.

Classe	Precision	Recall	F1-score	Supporto
Toxic	0.98	0.60	0.74	6090
Severe Toxic	0.30	0.27	0.28	367
Obscene	0.84	0.54	0.66	3691
Threat	0.00	0.00	0.00	211
Insult	0.75	0.47	0.58	3427
Identity Hate	0.71	0.11	0.18	712
Micro avg	0.857	0.511	0.640	

classi di tossicità e (2) lavorare sul pre-processing utilizzando Transfer Learning per ottenere feature più significative.
Il secondo esperimento è stato, infatti, caratterizzato da un processo di data augmentation, applicato sulle classi meno popolate, ovvero Severe Toxic e Threat, come descritto nel capitolo 3.4. Il modello appreso ha permesso di migliorare i risultati ottenuti nel primo esperimento, come è possibile osservare in Tabella 7.



(a) Loss. **(b)** Accuracy. **(c)** F1-score.

Figura 9: Andamento delle metriche principali durante il training della rete con data augmentation.

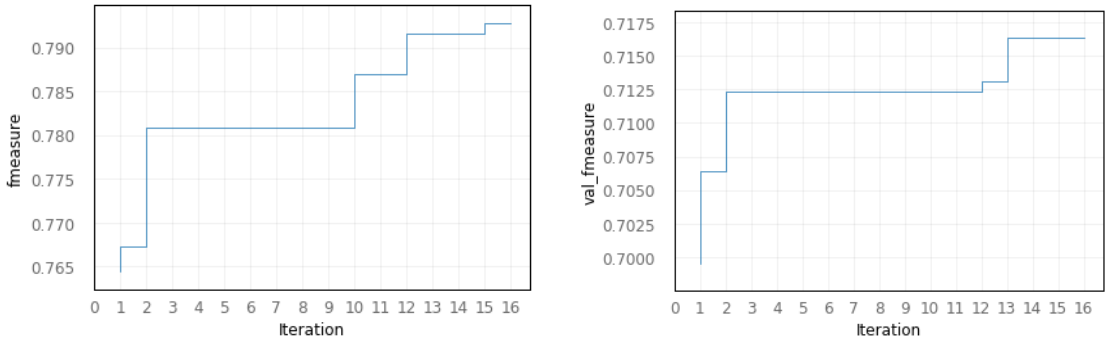
Le performance durante le 18 epoche della fase di apprendimento sono visibili in Figura 9. L'accuracy in training mostra un picco importante, sintomo di un possibile overfit nel caso in cui l'apprendimento fosse durato di più.

Tabella 7: Precision, Recall e F1-score per l'esperimento con Data Augmentation.

Classe	Precision	Recall	F1-score	Supporto
Toxic	0.98	0.67	0.79	6090
Severe Toxic	0.19	0.65	0.30	367
Obscene	0.80	0.61	0.69	3691
Threat	0.41	0.15	0.22	211
Insult	0.68	0.59	0.64	3427
Identity Hate	0.61	0.25	0.36	712
Micro avg	0.76	0.61	0.68	

Infine, il terzo esperimento è stato caratterizzato dall'approccio Feature Extraction descritto nella sezione 3.5. Questo modello è stato sottoposto alla fase di HPO

(sezione 3.6) e la libreria utilizzata ha permesso di evitare il training finale con i migliori iperparametri individuati, salvando direttamente il modello. Per questa ragione, non è possibile mostrare i grafici come nei due precedenti esperimenti. Tuttavia, è possibile osservare l'andamento dei valori di Best Seen per le metriche target F1 e Validation F1, in Figura 10.



(a) Best Seen F1-score. **(b)** Best Seen Validation F1-score.

Figura 10: Andamento del Best Seen durante il task di otpimization.

Le migliorie visibili graficamente, hanno infatti condotto a performance notevolmente migliorate sul Test set, come mostrato in Tabella 8.

Tabella 8: Precision, Recall e F1-score per l'esperimento con Feature Extraction e HPO.

Classe	Precision	Recall	F1-score	Supporto
Toxic	0.99	0.74	0.85	6090
Severe Toxic	0.41	0.27	0.32	367
Obscene	0.84	0.67	0.74	3691
Threat	0.76	0.36	0.49	211
Insult	0.79	0.62	0.69	3427
Identity Hate	0.75	0.32	0.45	712
Micro avg	0.877	0.656	0.751	

I risultati raggiunti in termini di Micro Average F1-score nel corso dei vari esperimenti sono, per semplicità visiva, riassunti in Tabella 9.

Tabella 9: Valori di Micro Average F1-score sul Test set nei vari esperimenti.

Esperimento 1	Esperimento 2	Esperimento 3
0.640	0.680	0.751

5 Discussione dei risultati

Come ampiamente discusso precedentemente, l'obiettivo di questo progetto consisteva nell'apprendere un modello in grado di classificare i commenti tossici secondo

sei diverse tipologie. Data la distribuzione delle classi, le metriche importanti da valutare sono Precision, Recall e F1-score. La Tabella 9 mostra come le soluzioni adottate abbiano permesso di migliorare i risultati, che tuttavia restano non pienamente soddisfacenti. Durante le varie analisi, una maggiore attenzione è stata posta sui valori di Recall: questa metrica fornisce un'idea su quanti commenti appartenenti ad una determinata classe *non* sono stati classificati come positivi, quando in realtà lo erano. Analizzando l'evoluzione dei valori di Recall nei tre esperimenti infatti, è possibile notare come il processo di oversampling sulla classe Severe Toxic abbia permesso di raggiungere i valori più alti. Questa considerazione potrebbe consentire performance migliori applicando oversampling su tutte le classi, ed applicando BERT come Feature Extractor. Infatti, l'ultimo esperimento ha permesso di ottenere i risultati migliori in termini di F1-score, prova delle capacità di discriminazione di BERT.

Approfondendo l'analisi, è possibile osservare come le classi di minoranza siano effettivamente le più complesse da apprendere, e quindi classificare. In accordo a quanto mostrato in Tabella 2, Threat e Severe Toxic sono le classi meno popolate nel Train set, seguite dalla classe Identity Hate. Queste proporzioni sono mantenute nel Test set e per questo motivo le scarse capacità di classificazione su queste classi non hanno portato grandi cali di performance in ambito micro average. E' bene precisare però che classi come Severe Toxic presentano in realtà un concetto astratto: quando un commento è semplicemente Toxic e quando diventa Severe? La classe Identity Hate è stata caratterizzata da buoni valori di Precision e bassa Recall durante tutti gli esperimenti. Questa caratteristica potrebbe indicare un *unintended bias* su parole specifiche indicanti etnie. Il fatto che sia non voluto però non implica che sia un problema, dato che ha permesso di migliorare la precisione.

Per quanto riguarda la classe più generica Toxic, le performance si sono rivelate molto buone in tutti gli esperimenti. Questa considerazione porta a spostare il problema sulle varie tipologie più astratte e, talvolta, interpretabili. Un modello il cui scopo fosse la classificazione soltanto in tossico/non tossico, avrebbe infatti performance elevate.

A conferma della opinabilità delle classificazioni, sono state svolte delle analisi ulteriori sui commenti classificati male dall'ultimo modello presentato. Queste analisi sono mostrate nel *terzo notebook* e hanno evidenziato come vi sia un problema sulle etichette dei commenti. In molti casi, le etichette sembrano essere state utilizzate a sproposito, classificando ad esempio come Identity Hate commenti che di odio razziale non ne hanno, ma ad esempio contengono l'uso di termini come Jew. Questi errori portano a pensare che l'etichettatura sia stata svolta da Macchine con pesanti *bias* su alcuni termini. Un'ulteriore caratteristica dei commenti è la presenza di insulti con lunghe ripetizioni di lettere per enfatizzare l'offesa. Il modello appreso nel terzo esperimento considerava questi, ad esempio, come Severe Toxic, quando la vera etichetta era una semplice Toxic. Potenzialmente, questa classificazione non è un errore, bensì un valore aggiunto. Le considerazioni espresse portano a pensare che il modello sviluppato, in realtà, funzioni meglio di quanto le performance mostrino.

Infine, è necessario considerare che BERT pone un vincolo sul numero di token

massimo nelle sequenze, ovvero 512. La presenza di commenti che superavano questa lunghezza è stata gestita semplicemente tagliando la coda di queste. Un approccio di questo genere può aver influito sulle performance, rischiando la perdita di informazione. Una soluzione più complessa a questo problema avrebbe preso in considerazione un preventivo task di text-summarization, per ridurre la dimensione.

6 Conclusioni

La classificazione di commenti tossici è sicuramente un task complesso, che richiede studi approfonditi sul problema e sulle sfumature linguistiche. Le soluzioni proposte hanno portato a risultati gradualmente sempre più soddisfacenti. L'utilizzo di uno dei migliori modelli transformer-based allo stato dell'arte come BERT, ha dimostrato una importante capacità di rappresentazione vettoriale, riuscendo ad apprendere diverse sfumature di tossicità. Avere a disposizione hardware più potente permetterebbe l'utilizzo del modello con un approccio fine-tuning e utilizzare deep neural network più avanzate, come ad esempio una LSTM, avrebbe portato molto probabilmente a risultati migliori.

Durante le analisi dei risultati sono emersi due problemi principali. Il primo riguarda il limite sulla dimensione delle sequenze di BERT, 512. Molti commenti presentavano più token rispetto a questo limite. La soluzione adottata in questo progetto è, probabilmente, la più semplice, ovvero tagliare la coda in eccesso, rischiando però di perdere informazione. Soluzioni più elaborate potrebbero considerare un task preventivo di text-summarization, e la successiva classificazione dei summary. Il secondo problema riscontrato riguarda le etichette di molti commenti tossici nel Test set. In particolare, l'erronea etichettatura dei commenti non ha permesso di ottenere una reale indicazione sulle performance dei modelli appresi. Inoltre, questo problema potrebbe aver causato un apprendimento non del tutto corretto, nel caso in cui anche il Train set fosse stato etichettato in egual modo.

Concludendo, uno sviluppo futuro possibile per approfondire lo studio preso in esame in questo progetto, potrebbe consistere nell'applicare il processo di Data Augmentation a tutte le classi di tossicità e successivamente applicare Transfer Learning con BERT.

Riferimenti bibliografici

[1] Kaggle. *Toxic Comment Classification Challenge*. URL: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge.

[2] Wikipedia. *Wikipedia Talk pages*. URL: https://en.wikipedia.org/wiki/Help:Talk_pages.

[3] Jeffrey Pennington, Richard Socher e Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[4] Sashank J. Reddi, Satyen Kale e Sanjiv Kumar. "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ.

[5] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[6] Han Xiao. *bert-as-service*. https://github.com/hanxiao/bert-as-service. 2018.

[7] Autonomio. *Talos*. 2019. URL: http://github.com/autonomio/talos.