

**UNIVERSITÀ DEGLI STUDI DI BARI   
“ALDO MORO”**

Dipartimento di Informatica

Corso di Laurea in Informatica

*Tesi di Laurea in Modelli per la Sicurezza delle Applicazioni*

**Titolo Tesi**

**Creazione di un modello di analisi testuale per i commenti aggressivi nei social.**

*Relatore*:  
Prof. Donato Impedovo

*Correlatore*:  
Dott. Vincenzo Gattulli

*Laureando*:  
 Pasquale Santoro

Anno Accademico 2021/22

*Alla mia famiglia.*

Pasquale Santoro

**Abstract**

La crescita esponenziale del web e dei social media avvenuta nel XXI secolo ha rivoluzionato il modo di comunicare fra i giovani e non. Essi sono diventati dei veri e propri luoghi virtuali dove si instaurano amicizie e relazioni sociali, un luogo dove vengono meno le “barriere comunicative” della realtà. Tuttavia, questi spazi virtuali non sono immuni agli aspetti negativi presenti nella realtà, come il bullismo.

Obiettivo di questa tesi è di trovare un sistema di analisi testuale che sappia identificare i commenti negativi e le aggressioni testuali nei social media, esplorando le relazioni tra commenti negativi e aggressioni, ed infine verificare una correlazione tra cyberbullismo e commenti sarcastici.

Sono stati eseguiti tre esperimenti:

* La prima sperimentazione ha l’obiettivo di trovare la miglior fase di *preprocessing* e *feature* *extraction*, supportata dalle migliori ricerche allo stato dell’arte, che sappia identificare i commenti contenenti cyberbullismo e commenti dal sentimento negativo. La miglior combinazione risulterà essere composta da standard *preprocessing*, GLOVE embedding e BiLSTM.
* Nel secondo esperimento si cercherà di chiarire se il sistema di identificazione ottenuto dal primo esperimento possa beneficiare di un addestramento cross-dataset con dati semantici differenti, e se esiste una correlazione tra commenti negativi e aggressivi: relazione che verrà parzialmente trovata.
* L’ultimo esperimento esplorerà la correlazione tra commenti sarcastici e commenti contenenti cyberbullismo. Seppur nel mondo reale il sarcasmo è una delle tattiche preferite dagli aggressori, nel mondo virtuale sembra che essi preferiscano degli attacchi più diretti.

**Indice**

Sommario

[Capitolo 1 – Introduzione 9](#_Toc76291810)

[1.1 Cyberbullismo 9](#_Toc76291811)

[1.2 Sarcasmo 12](#_Toc76291812)

[Capitolo 2 – Stato dell’arte 14](#_Toc76291813)

[2.1 Identificazione di commenti aggressivi 14](#_Toc76291814)

[2.1.1 Approccio Multimediale 16](#_Toc76291815)

[2.1.2 Approccio testuale 17](#_Toc76291816)

[2.2 Identificazione del cyberbullo 19](#_Toc76291817)

[2.3 Identificazione del sarcasmo 21](#_Toc76291818)

[Capitolo 3 – Progettazione 23](#_Toc76291819)

[3.1 Introduzione al machine learning 23](#_Toc76291820)

[3.2 I paradigmi dell’apprendimento automatico 24](#_Toc76291821)

[3.3 Shallow learning 25](#_Toc76291822)

[3.3.1 Random Forest 25](#_Toc76291823)

[3.3.2 SVM 25](#_Toc76291824)

[3.3.3 Logistic Regression 26](#_Toc76291825)

[3.4 Deep Learning 27](#_Toc76291826)

[3.4.1 Reti Feed-Forward 29](#_Toc76291827)

[3.4.2 Reti Ricorenti 31](#_Toc76291828)

[3.5 Dataset 34](#_Toc76291829)

[3.6 Metriche utilizzate 34](#_Toc76291830)

[3.7 Librerie utilizzate 35](#_Toc76291831)

[3.8 Primo esperimento 36](#_Toc76291832)

[3.9 Secondo esperimento 42](#_Toc76291833)

[3.10 Terzo esperimento 43](#_Toc76291834)

[Capitolo 4 – Sperimentazioni 44](#_Toc76291835)

[4.1 Primo esperimento 44](#_Toc76291836)

[4.2 Secondo esperimento 47](#_Toc76291837)

[4.3 Terzo esperimento 47](#_Toc76291838)

[Capitolo 5 – Conclusioni 49](#_Toc76291839)

[Riferimenti 51](#_Toc76291840)

# Capitolo 1 – Introduzione

Introduzione

## 1.1 Cyberbullismo

Il primo a coniare il termine cyberbullismo fu un educatore canadese chiamato Bill Belsey, già nel lontano 2002, quando ancora il concetto di social era ancora agli albori. Tuttavia la definizione più utilizzata è quella di Peter K. Smith et al. [1] che definisce il cyberbullismo come un atto aggressivo e intenzionale, condotto da un individuo o gruppo di individui, usando varie forme di contatto elettronico e ripetuto nel corso del tempo contro una vittima.

Se i social ci permettono di comunicare e condividere pensieri, opinioni e anche emozioni, dall’altra parte il rischio di essere vittima di cyberbulli per i ragazzi in età adolescenziale cresce in maniera vertiginosa.

Il bullismo “tradizionale” colpisce principalmente negli ambienti come le scuole e i piccoli contesti sociali, tuttavia il cyberbullismo insieme alla possibilità di anonimato rendono i social un “terreno fertile” per le aggressioni, andando a colpire potenzialmente molte più vittime appartenenti a categorie sociali, nazionalità ed età differenti. [2] Gli effetti psicologici sono, al pari del bullismo tradizionale devastanti: secondo Kowalski et al. [3] il cyberbullismo può causare ansia, depressione e nei casi più gravi anche il suicidio.

Innumerevoli sono le iniziative condotte contro questo fenomeno: già nel 2004 la commissione Europa istituì il “Safer Internet Day”, una giornata internazionale per la tutela dei giovani sull’uso del web e dei social. In Italia, secondo le ultime ricerche condotte nel 2020 dall’osservatorio Indifesa, circa il 61% di ragazzi con età compresa dai 13 ai 23 anni ammettono di essere stati vittime di cyberbullismo almeno una volta, numeri che impongono delle urgenti contromisure.

Inoltre, secondo Jean-Baptiste et al. [4] una maggiore esposizione sui social aumenta la probabilità di subire delle cyber-aggressioni; considerando che gli adolescenti passano più di 4 ore al giorno sui social, secondo i dati raccolti dal Movimento Etico Digitale, è facile intuire che la situazione delle giovani vittime non può che peggiorare. Questi dati però non tengono conto dell’attuale situazione sanitaria aggravata dai vari lockdown avvenuti nel 2020 per contrastare la pandemia di COVID-19, che si traducono in un ulteriore abuso degli adolescenti dei social media.

Le piattaforme con più aggressioni, secondo il “The annual bullying survey” si confermano Facebbok e Instagram, in assoluto i social più utilizzati dagli adolescenti (figura 1)

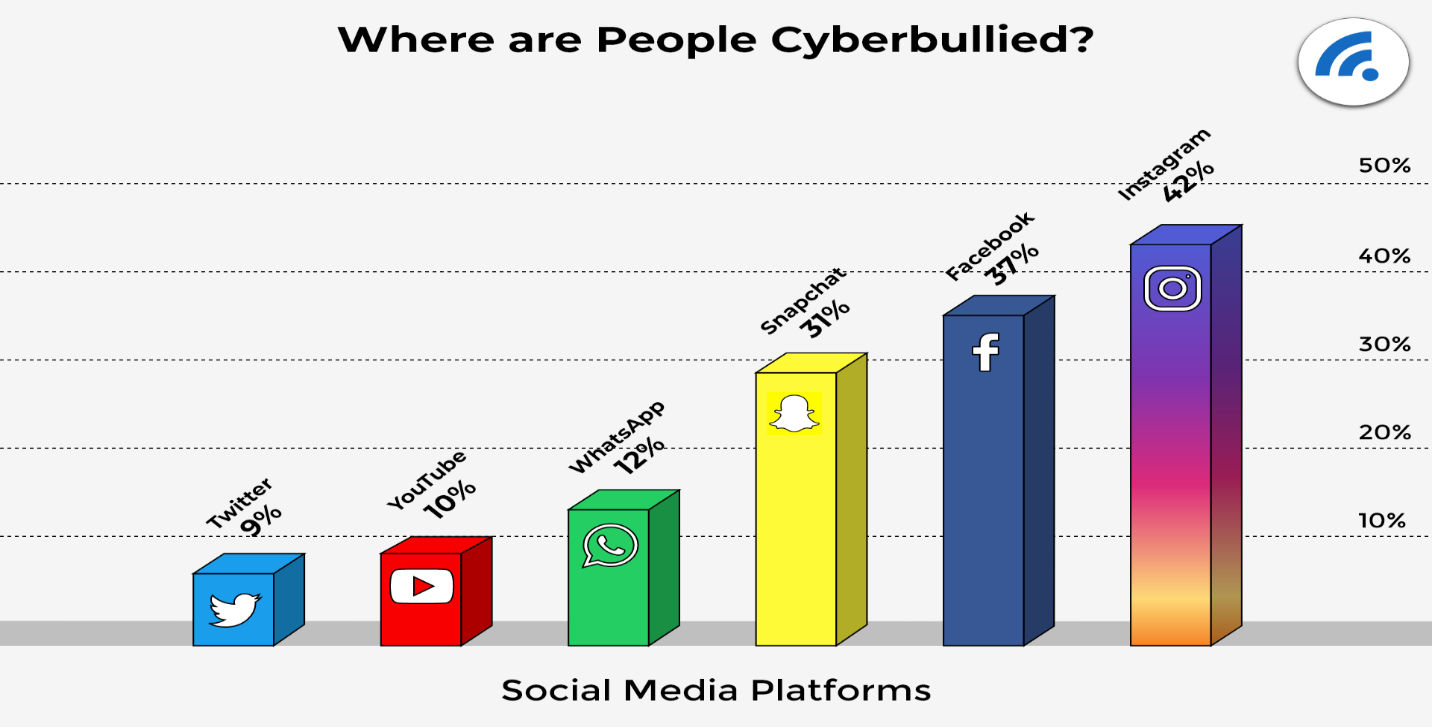


Figura 1

Come nel bullismo tradizionale, si può essere vittima di vari tipi di cyberbullismo:

* *Flaming:* con questo termine si indica l’uso di messaggi violenti, volgari o aggressivi che hanno lo scopo di suscitare “battaglie” tra una o più persone. Lo scontro avviene solitamente ad armi pari, in quanto tutti i partecipanti rispondono in modo aggressivo.E’ possibile trovarne degli esempi in quasi tutte le communities di gaming, soprattutto da parte del sesso maschile.
* *Harassment:* consiste nell’utilizzo messaggi, commenti e aggressioni virtuali ripetutamente nel tempo contro una vittima. Vi è quindi uno sbilanciamento di potere tra vittime e l’aggressore. I bulli di questo tipo commettono anche un reato penale.
* *CyberStalking:* declinazione dell’*harassment* in cui le vittime temono anche per la loro salute fisica. Si riscontra facilmente nelle relazione sentimentali interrotte. L’aggressore, oltre ad aggredire la vittima, minaccia spesso la condivisone di contenuti intimi.
* *Denigration:* con questo termine si indica la pratica della diffamazione della vittima attraverso pettegolezzi o la diffusione di video o foto on-line. I cyberbulli possono anche pubblicare delle foto non veritiere, ad esempio effettuando dei fotomontaggi.In questo caso gli spettatori sono degli elementi passivi, ma che possono diventare attivi scaricando o condividendo a loro volta questi contenuti. Esempi comuni si possono ricontrare tra le relazioni amorose, o nel rapporto tra studenti e professori.
* *Impersonation:* avviene quando una persona riesce ad accedere agli account personali dei social di una vittima, e li sfrutta per danneggiare la reputazione o le amicizie. A differenza delle altre aggressioni potrebbe passare del tempo prima che la vittima se ne renda conto.
* *Outing and trickery:* con outing si intende la pratica in cui l’aggressore, dopo aver ricevuto delle confidenze di un coetaneo attraverso sms o chat, decide di diffonderle sul web. Con il secondo termine si indica la situazione in cui l’aggressore spinge la vittima a condividere segreti o informazioni imbarazzanti su se stesso, anche sotto minaccia. A differenza delle tipologie precedenti, l’aggressore ha inizialmente un rapporto bilanciato (almeno apparentemente) con la vittima, assumendo il ruolo di carnefice successivamente.
* *Exclusion:* l’aggressore decide di escludere un suo coetaneo da una chat di gruppo, videogame, o ambiente virtuale. Questa pratica viene comunemente chiamata “ban”.
* *Happy slapping:* con questo termine si indica la pratica di condividere video o foto di aggressioni fisiche sul web. Spesso i contenuti di questo genere diventano virali, attirando commenti o interazioni. E’ molto comune trovare del cyberbullismo di questo genere sulla piattaforma di Facebook, in quanto offre la possibilità di condividere video o foto con dei sistemi di censura molto spesso fallaci.

Riconoscere e classificare correttamente il cyberbullismo all’interno del mondo digitale, tramite le ultime tecniche dell’intelligenza artificiale, rappresenta un primo step nella risoluzione del problema.

Questa tesi nasce principalmente con l’obiettivo di trovare il migliore sistema, supportato dalle ricerche dello stato dell’arte, per l’identificazione di commenti aggressivi e contenuti testuali dal significato negativo, come ad esempio il seguente tweet preso dal dataset Cyber-troll, oggetto di varie sperimentazioni all’interno della tesi: “*WTF are you talking about Men? No men thats not a menage that's just gay*”, letteralmente : “Di cosa state parlando? Non c’è niente di virile in questo, è solo gay”. Si è deciso infatti di studiare insieme questi due tipi di contenuti, per investigare una eventuale correlazione tra di essi all’interno dell’ambito della *sentiment* *analysis*.

Inoltre verranno utilizzati dataset bilanciati e non. Questo poiché in una situazione reale è difficile raccogliere abbastanza dati da creare un dataset appropriato per la corretta classificazione: questo studio pone l’enfasi su questo aspetto, in modo da trovare un sistema che si adatti bene anche in queste situazioni molto comuni ma poco studiate, a causa dell’accuratezza minore che implica l’utilizzo di un dataset estremamente sbilanciato. Queste problematiche verranno affrontate nel corso primo esperimento.

La seconda domanda su cui si basa questo lavoro è la seguente: i modelli di apprendimento possono beneficiare di un addestramento che comprende diverse fonti semanticamente simili? Ma soprattutto esiste una correlazione tra i commenti dal sentimento negativo e il cyberbullismo? Per commenti dal sentimento negativo si intende qualunque testo che abbia un significato negativo nel significato più ampio del termine, come ad esempio la seguente frase, presa dal dataset ACLL: “*This is one of the dumbest films, I've ever seen. It rips off nearly ever type of thriller and manages to make a mess of them all.” .* ACLL è uno dei due dataset etichettati con sentimento positivo e negativo che verranno utilizzati in questo lavoro, e raccoglie circa ventimila recensioni di film. Il secondo esperimento di questa tesi cercherà di dare delle risposte a queste domande.

## 1.2 Sarcasmo

Essendo i social un “prolungamento” della vita reale e delle dinamiche sociali, è facile imbattersi in forme di ironie e sarcasmo all’interno di essi.

Le differenze tra i due sono sottili, ma fortemente discriminatorie. L’ironia consiste nell'affermare il contrario di ciò che si pensa con lo scopo di ridicolizzare o sottolineare concetti per provocare una risata[[1]](#footnote-1),vi è quindi un intento positivo. Un esempio di ironia può essere l’esclamazione della frase: “Hai avuto un’idea geniale!” nel caso di una decisione che ha portato effetti catastrofici.

Il sarcasmo viene invece definito dal dizionario di Oxford[[2]](#footnote-2) come l’utilizzo di parole che hanno un significato opposto a quello che si intende dire, per mettere a disagio qualcuno o per offendere velatamente qualcuno. In esso vi è un intento beffardo. L’identificazione del sarcasmo è una delle sfide aperte strettamente connesse al bullismo e cyberbullismo. Rosenthal et al. [5] mostrano come le tecniche classiche di *sentiment analysis* si dimostrano inefficaci nell’identificazione di queste figure retoriche.

Il sarcasmo nei social inoltre, non è solo testuale. Sulle piattaforme più famose, quali Facebook e Instagram, infuriano la creazione e la condivisione dei meme, dei contenuti multimediali che racchiudono testo e immagini in forma ironica o sarcastica(fig. 2).

Immagine che contiene testo, persona, cravatta

Descrizione generata automaticamente

Figura 2. Meme sarcastico che potrebbe risultare offensivo in base al contesto

Inoltre esso è utilizzato molto spesso anche dai bulli negli ambienti reali, dato che offre una perfetta occasione per mascherare un offesa come semplice scherzo.

Seguendo il lavoro di Zheng Lin Chia et al. [6] si verificherà una possibile correlazione tra il cyberbullismo e il sarcasmo, effettuando una sperimentazione cross-dataset. Questa correlazione sarà oggetto del terzo e ultimo esperimento.

La tesi sarà organizzata nel seguente modo: nel secondo capitolo verranno descritti gli studi più significativi dello stato dell’arte; nel terzo capitolo verrà descritta la progettazione degli esperimenti e

i concetti teorici utilizzati; il quarto capitolo conterrà la descrizione dei tre esperimenti con i relativi risultati; infine, nel quinto capitolo verranno tratte le conclusioni e i possibili sviluppi futuri.

Questo lavoro inoltre è stato sviluppato per PRIN2017 – BullyBuster project – un framework per l’identificazione di bullismo e cyberbullismo attraverso l’intelligenza artificiali e metodi algoritmici.

# Capitolo 2 – Stato dell’arte

Il tema del cyberbullismo e del riconoscimento dei commenti aggressivi sui social si sta sempre più affermando all’interno della comunità scientifica, grazie anche all’utilizzo massiccio di queste tecniche da parte dei social più popolari.

In questa sezione si introdurranno gli studi più significativi in questo ambito, che costituiscono il punto di partenza per le varie sperimentazioni.

## 2.1 Identificazione di commenti aggressivi

Per quanto riguarda l’identificazione di cyberbullismo le ricerche attuali si possono dividere su due fronti: il riconoscimento dei commenti offensivi dal testo o dalle immagini contenenti didascalie(fig. 3).

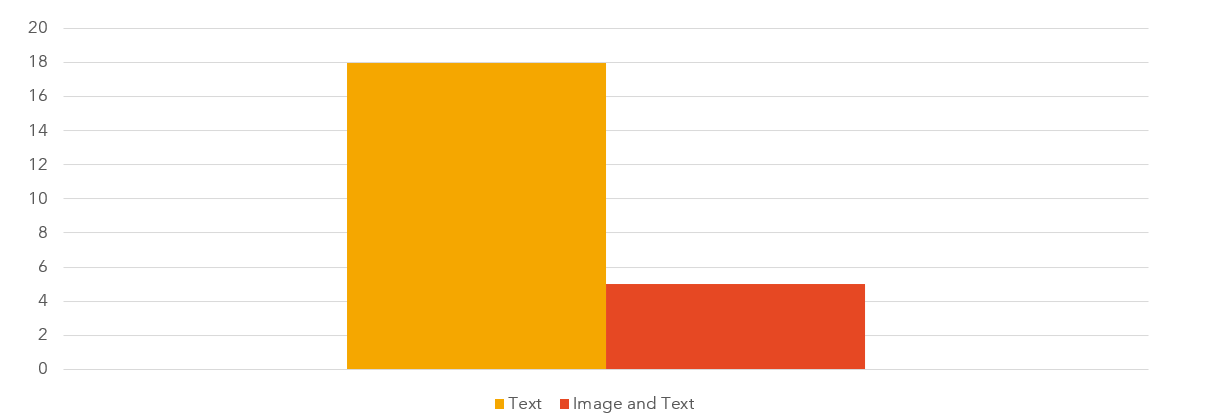


Figura 3. Single vs Multimodal approach nelle ricerche selezionate

Per quanto riguarda le sorgenti da cui sono stati creati i dataset utilizzati nello stato dell’arte, Twitter si conferma il social più studiato, seguito da FormSprings, un social basato sul meccanismo di domande e risposte.

I motivi per cui Twitter è il social più studiato sono principalmente tre:

* Enorme quantità di contenuti;
* Facilità nell’effettuare lo *scraping* dei contenuti attraverso le API ufficiali;
* Possibilità di raccogliere delle network feature, quali numero di amici, follower, iscrizione etc.

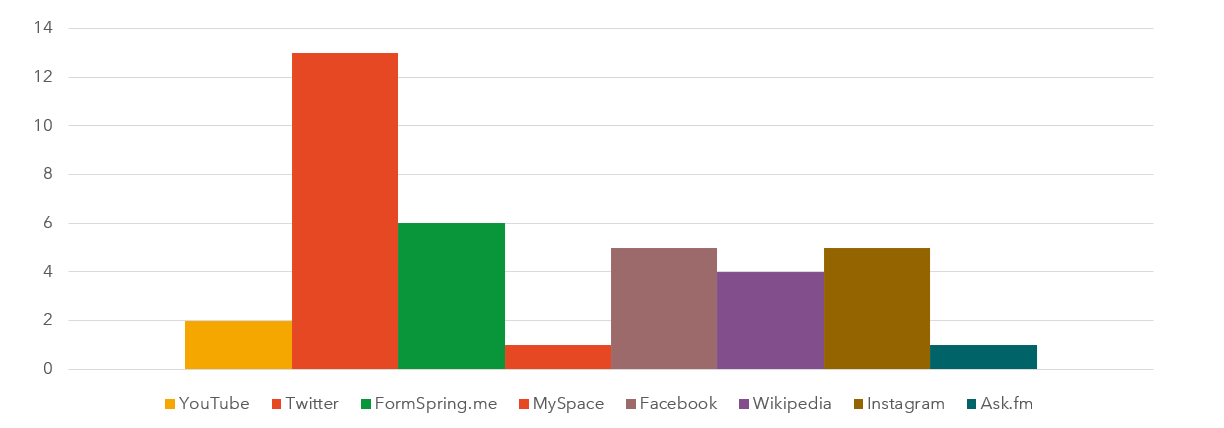


Figura 4. Sorgenti coinvolte negli studi

Per quanto riguarda i tipi di classificatori utilizzati, sia i modelli *shallow* *learning* che *deep* *learning* vengono impiegati con ottimi risultati.

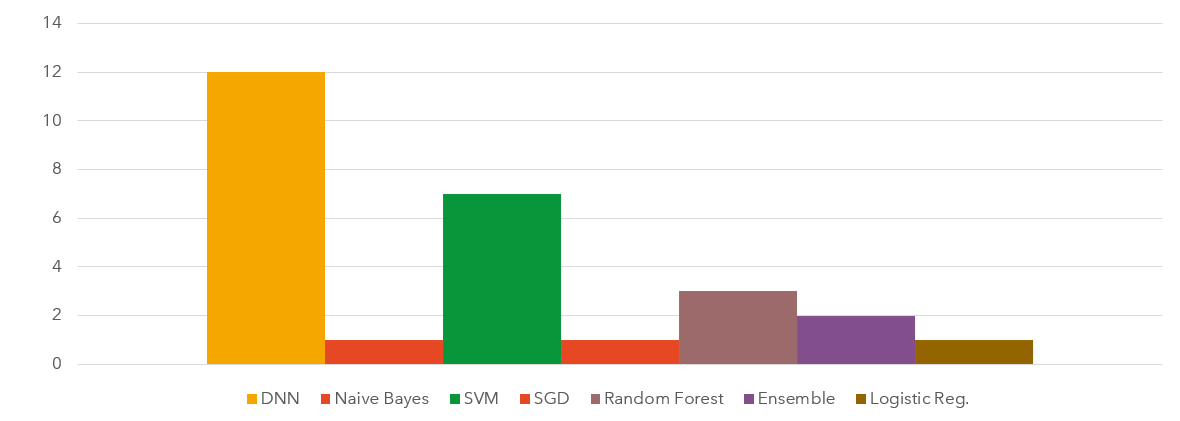


Figura 5. Classificatori shallow a confronto

Per i tipi di reti neurali, nello stato dell’arte vengono impiegate maggiormente reti a convoluzione e reti ricorrenti. Quest’ultime, data le loro caratteristiche si rendono particolarmente adatte nei casi in cui è utile decifrare il contesto di una frase.

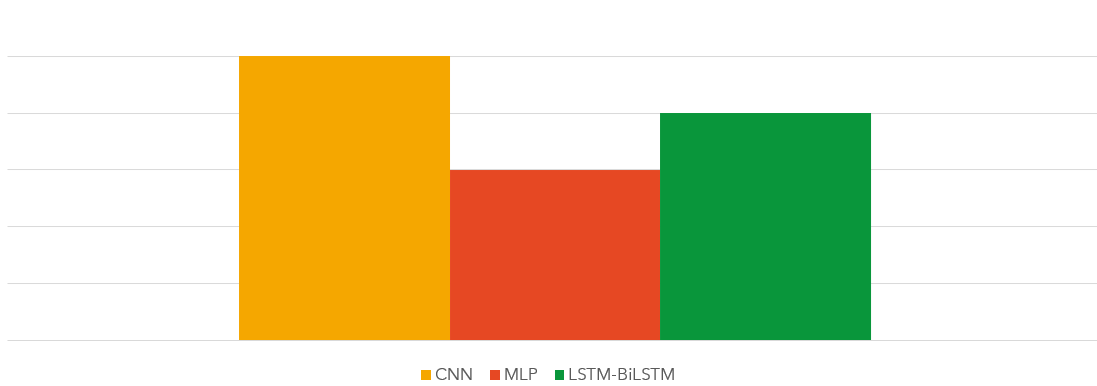


Figura 6. Tipi di reti neurali

Di seguito verrà presentata una panoramica sui migliori studi dello stato dell’arte, divisi per approccio multimediale, testuale , profilazione dei cyberbulli e identificazione del sarcasmo.

### 2.1.1 Approccio Multimediale

Di particolare interesse sono i lavori di K. Kumari,J. P. Singh et al. [7] [8] . Nel primo studio gli autori implementano una CNN per il riconoscimento di immagini con didascalie contenenti cyberbullismo(fig. 7)

Immagine che contiene testo

Descrizione generata automaticamente

Figura 7.

Come *feature extraction* viene utilizzato il tf-idf per il testo, in modo da creare una matrice bidimensionale. In questo modo viene creato un sistema unico per il testo e per le immagini, con un F-Score del 71% nella classe contente media aggressivi.

Nel secondo studio gli autori ampliano la ricerca precedente utilizzando due diversi sistemi: una rete neurale pre-addestrata per il riconoscimento delle immagini, chiamata VGG-16, e una CNN per il riconoscimento del testo. Le feature estratte dalle due reti neurali vengono combinate tra di loro e un algoritmo genetico seleziona le migliori feature, ottenendo un F-Score del 74%

Un approccio simile viene seguito da Akshi Kuami et all. [9]. Essi costruiscono un sistema all-in-one chiamato CapsNet-ConvNet, consistente in una rete a convoluzione per la classificazione delle immagini(ConvNet), e un *capsule* *network* con routing dinamico per la classificazione del testo(CapsNet). Prima di essere processati i contenuti multimediali vengono splittati attraverso Google Lens di Google Photos app, e dopo essere stati processati le feature raggiungono una MLP per la classificazione finale.

Testando il sistema su un dataset contente 10,000 commenti, immagini e contenuti multimediali ottengono un punteggio ROC di 0.98.

### 2.1.2 Approccio testuale

Uno dei punti di partenza per questa tesi è il lavoro di Sweta Agrawal et al. [10], un esperimento multi-piattaforma che analizza le prestazioni di quattro tipi di reti neurali profonde con diversi tipi di embedding nel riconoscimento di commenti aggressivi, tra cui la CNN e la BiLSTM utilizzate negli esperimenti di questa tesi.

Essi comparano le prestazioni di quattro tipi di neurali profonde(CNN, BiLSTM e BiLSTM con attenzione), utilizzando i dataset ottenuti dalle sorgenti di FormSprings, Twitter e Wikipedia effettuando un test con *oversampling* e un test senza. Il risultato è un aumento dell’accuratezza per i dataset che hanno ricevuto *oversampling,* in particolare per i dataset molto sbilanciati come Formsprings. Gli autori inoltre sperimentano che nella maggior parte dei casi, un oversampling con un fattore 3x è sufficiente per ottenere ottime performance.

Nella sperimentazione successiva testano 3 tecniche di word embedding: random, GLOVE, e SSWE. L’ultima tecnica è un tipo di word embedding che incorpora in esso anche il sentimento delle parole [11]. I risultati mostrano che in assenza di oversampling, e quindi con dataset sbilanciati vi è una differenza di performance tra le architetture più semplici (CNN) e quelle più complesse (BiLSTM con attenzione). Tuttavia tutte le tecniche di word embedding sono valide nel caso di dataset bilanciati.

Nell’ultima sperimentazione utilizzano il *transfer* *learning* per verificare se la conoscenza acquisita dalle reti neurali su un dataset possa essere utilizzata per aumentare le performance nel riconoscimento di cyberbullismo sugli altri dataset. Sperimentano tre tipi di tranfer learning:

* *Complete Transfer Learning*, ovvero addestramento su un dataset e test su un altro dataset, senza extra training. I risultati mostrano una bassa *recall,* ad indicare una natura differente di cyberbullismo tra i dataset di Twitter, FormSprings e Wikipedia. Tuttavia gli ultimi due mostrano delle analogie, dovute al fatto che sono entrambi *task* *oriented* eorganizzate in domande e risposte, a differenza di Twitter che non copre un particolare argomento e non consente l’anonimato.
* *Feature Level Transfer Learning.* Con questa tecnica un modello viene addestrato su un dataset, e soltanto il word embedding appreso viene trasferito su un altro dataset e riaddestrato. Ne consegue un aumento della *precision* e del *recall,* che indica come il word embedding addestrato sia essenziale per il trasferimento di conoscenza tra dataset.
* *Model Level Transfer Learning*. Con questa tecnica vengono trasferiti sia il word embedding e sia i pesi della reti neurale, tuttavia non ci sono miglioramenti rispetto la tecnica del Feature Level Transfer Learning.

Lo studio di Mara Dadvar et al. [12] riprende e conferma gli esperimenti dello studio precedente. Lavorando sugli stessi dataset(FormSprings, Twitter e Wikipedia) effettua una sperimentazione cross-dataset utilizzando la tecnica del transfer learning. In particolare applica la conoscenza imparata sui dataset precedenti su un nuovo dataset composto da commenti presi da YouTube, creato da [13]. Addestrando sul dataset di FormSpring oversamplato, ottiene un F1-Score nel test su YouTube di 0.30, 0.62, 0.76 rispettivamente con le tecniche di Complete, Feature Level e Model Level Transfert learning.

Un’altra conferma della bontà delle architetture deep proviene da V. Banerjee, Juy Telavane et al. [14], in particolare dall’utilizzo di una CNN con GLOVE embedding, testato su un dataset di circa 70mila tweets, ottenendo un’ accuratezza del 94%.

V. S. Chavan e Shylaja S. S. [15] propongono due nuove ipotesi: l’utilizzo di n-grammi e la conta dei pronomi, come “tu” e “voi”. Essi utilizzando il tf-idf come *feature extraction* e il metodo del Chi-quadro come *feature selection,* su un dataset recuperato da Kaggle[[3]](#footnote-3). Partendo da una *baseline* del 83% di accuratezza con *tf-idf* e regressione logistica, le due nuove ipotesi portano un aumento del 3% sullo stesso dataset.

Sulla piattaforma Facebook K. Gorro, M. Sabellano et. al [16] sperimenta una precisione dell’85% utilizzando tf-idf come feature extraction e Support Vector Machine. Risultati simili sono ottenuti da T. Davidson et al. [17] sulla piattaforma Twitter utilizzando le stesse feature extraction e lo stesso modello dello studio precedente, ma con l’aggiunta del Porter stemming nella fase di *preprocessing.*

Il lavoro di [18] consiste nello sperimentare diversi tipi di *preprocessing* e *feature selection* sulle piattaforme di FormSprings, YouTube e MySpace ottenendo un F-Score del 92% attraverso una rete neurale e tf-idf sul dataset FormSprings con *oversampling*.

In particolare rileva che la miglior combinazione di *preprocessing* per MySpace è rappresentata da Uppercase, Stemming e rimozione di *stopword*; per FormSprings è rappresentata da Lowercasing, Stemming e rimozione di *stopword*; mentre YouTube ha dato migliori risultati con l’utilizzo del solo Lowercasing.

Per quanto riguarda i metodi di feature selection, ne sperimenta quattro: Chi2, MRMR, ReliefF, SVM-RFE. Gli autori rilevano che in generale, utilizzando un metodo di feature selection vi è una diminuzione dell’accuratezza, a scapito di una maggior velocità di classificazione.

Particolarmente interessante risulta essere lo studio di S. Sadiq, A. Mehmood et al. [19] che dimostra come alcune architetture deep possano performare meglio utilizzando *tf-idf* in termini di *unigram* e *bigram* come *feature extraction*, in contrapposizione al word embedding. Essi utilizzando il dataset Cyber-troll, mettono a paragone tre reti neurali con word embedding (CNN-LSTM, CNN-BiLSTM, MLP) e una MLP da loro proposta che utilizza il tf-idf in termini di *unigram* e *bigram* come feature extraction, che risulterà essere la migliore. Tuttavia, a seguito di una fase di standard *preprocessing* utilizzano un metodo di feature selection per prevenire il problema dall’*overfitting*.

Sulla scia del *tf-idf* come feature extraction risalta lo studio di S. Muhamad, L. Ashanti et al. [20] che ottiene un’accuratezza del 99% utilizzando una Support Vector Machine, su un dataset contente conversazioni recuperati da FormSpring. Essi dimostrano che l’applicazione di n-grammi, da 2-gram a 5-gram, aumenta l’accuratezza nell’identificazione di commenti aggressivi. Inoltre rilevano che nella classificazione binaria, i kernel *linear*, *poly*, *sigmoid* presentano differenze minime, mentre utilizzando il kernel *rbf* vi è un decremento notevole della performance.

Diversi studi sono stati compiuti sulla piattaforma Wikipedia, attingendo ai commenti presenti all’interno di essa. Tra questi studi, M. Gadae k. et al. [21] paragonano Random Forest con tf-idf ad un architettura deep composta da LSTM e CNN, con word embedding. I modelli ottengono un F-score di 42.6% e 44% rispettivamente.

Le basi per la seconda sperimentazione vengono poste da M. Mahat [22]. L’autore implementa un addestramento cross-dataset, sfruttando una LSTM con word embedding sulle piattaforme di Twitter, Wikipedia e FormSprings ottenendo un’accuratezza del 77%, superando lo stato dell’arte rappresentato da Whittaker e M. Kowalski [23]

## 2.2 Identificazione del cyberbullo

Se l’identificazione di commenti contenti cyberbullismo risulta essere molto approfondita nello stato dell’arte, lo stesso non si può dire sull’identificazione del cyberbullo attraverso i social. Questo è dovuto al fatto che per riconoscere un cyberbullo sono necessarie delle feature aggiuntive che riguardano il suo comportamento, come numero dei post, pagine o gruppi seguiti nel caso di Facebook, numero di messaggi inviati etc. Informazioni che ovviamente non vengono rilasciate liberamente dai social, rendendo difficile lo sviluppo della ricerca.

L’unico social che mette a disposizione alcune di queste informazioni è Twitter, tramite la sua API ufficiale. Per questa ragione verranno presentati di seguito due di queste ricerche su questa tematica, sviluppate sul medesimo social.

Lo studio più citato riguarda il lavoro di Despoina Chatzakou et al. [24]. Essi considerano e analizzano più messaggi per ogni utente, raccolti in un certo intervallo di tempo. Lo scopo è quello di categorizzare efficacemente gli utenti in:

* Aggressori: chiunque abbia condiviso o scritto un contenuto con lo scopo di offendere qualcuno o di incitare all’odio;
* Bulli: chiunque abbia condiviso o scritto ripetutamente (>2) un contenuto offensivo sullo stesso argomento, contro lo stesso gruppo di persone o singolo.
* Spammer: chiunque condivida ripetutamente contenuti in maniera sospetta, con lo scopo di vendere qualcosa o tentativi di phishing.

Per la classificazione vengono utilizzate feature di tipo *user*, *textual* and network *based*. Con feature *user-based* si intendono tutte quelle informazioni come il numero dei post pubblicati, l’anzianità del profilo, se è un profilo verificato, le liste a cui è iscritto, se ha un’immagine del profilo etc. Nel ramo delle text-feature si utilizza invece il word embedding, sentiment score, numeri delle lettere in maiuscolo, hastag utilizzati. Tuttavia la gran parte delle informazioni discriminatorie derivano dalle network-feature, ovvero il numero di amici, il numero di *followers,* popolarità etc.

Utilizzando il classificatore Random Forest, ottengono una precisione media del 72% nella classificazione su quattro classi, mentre classificando solo su aggressori, bulli e utenti normali la precisione arriva 90%. Il motivo di questa differenza di accuratezza risiede nella difficolta nel distinguere gli utenti aggressivi dagli utenti spammer.

Essi arrivano alle seguenti conclusioni:

* I cyberbulli hanno meno amici/follower e partecipano in poche communities. Essi non condividono molti contenuti, ma quando cominciano a postare lo fanno con una frequenza superiore a quella degli utenti normali, utilizzando più hashtag e più URL.
* Gli utenti aggressivi sono i più difficile da etichettare, dato che alcune volte agiscono da utenti normali e altre volte da cyberbulli, anche se i loro post non sono cosi negativi come quelli dei bulli. In termini di numeri di follower e numeri di amici presentano similarità con gli utenti *spammer.*

Un approccio simile viene utilizzato da R. Ramesh et al. [25]. Essi utilizzano il tf-idf *come feature extraction* e SVM, limitandosi ad analizzare solo le *textual feature*.

In particolare se il tweet di un utente analizzato risulta essere classificato come aggressivo, vengono prelevati altri 10 tweet dello stesso utente e classificati. Se in media risultano essere cyber-aggressivi l’utente viene classificato come cyberbullo. Questo approccio può essere replicato più facilmente su altri social, date le minori informazioni richieste, ma ottiene una performance più bassa del 20% rispetto allo studio precedente.

## 2.3 Identificazione del sarcasmo

L’identificazione del sarcasmo risulta essere una delle sfide più interessanti e difficili degli ultimi anni. Se è relativamente facile identificare dei messaggi aggressivi, il riconoscimento del sarcasmo non può prescindere da un’ottima conoscenza del contesto della frase. Una qualunque frase ironica o sarcastica infatti, può essere positiva o negativa in base al rapporto tra mittente e destinatario.

Dal lavoro di P. Verma [26] risulta che per la *detection* possono essere adottate vari tipi di approcci, ad esempio basati sul lessico, o sul machine learning. Tuttavia le performance migliori vengono raggiunte combinando vari tipi di architetture deep, come dimostrato da L. H. Son et all. [27]. Essi implementato una BiLSTM con layer di attenzione insieme a una CNN, ottenendo un F-score dell’88% su un dataset di random tweets.

Il lavoro di Z. Chia, M. Ptaszynski et al. [6] pone le basi per la terza sperimentazione di questa tesi.

Essi nel primo esperimento testano sette tipi di preprocessing, per trovare la miglior combinazione nell’identificazione del sarcasmo. Gli esperimenti vengono compiuti sul dataset creato da A. Ghosh e T. Veale [28].

In particolare sperimentano le seguenti combinazioni:

1. Preprocessing di base;
2. URL, tag e hashtag etichettati;
3. Stemming Porter;
4. Rimozione di *stopword*;
5. Stemming e rimozione di *stopword*;
6. POS tag.
7. Rimozione di username, URL e hashtag, che non verranno etichettati.

Ad esempio il seguente tweet:

“@Alex I am loving Monday morning! #sarcasm [www.facebook.com](http://www.facebook.com) 😊”

Nelle varie combinazioni diventerà:

1. \_tagged\_ i am loving monday morning! #sarcasm \_url\_ :smiley\_face:
2. \_tagged\_ i am loving monday morning! \_hashtag \_url\_ :smiley\_face:
3. \_tagged\_ i am love monday morning! \_hashtag \_url\_ :smiley\_face:
4. \_tagged\_ loving monday morning! \_hashtag \_url\_ :smiley\_face:
5. \_tagged\_ love monday morning! \_hashtag \_url\_ :smiley\_face:
6. \_tagged\_ PRP\_i VBP\_am VGB\_loving NN\_monday NN\_morning!\_hashtag \_url\_ :smiley\_face:
7. I am loving monday morning! :smiley\_face:

Ogni combinazione di preprocessing è stata testata utilizzando una rete convoluzionale, e come metodo di feature extraction è stato utilizzato il tf-idf insieme al word embedding. A seguito di ciò la miglior combinazione per l’identificazione del sarcasmo risulta essere la 1) e la 2)

Successivamente gli autori cercano di verificare una correlazione tra sarcasmo e cyberbullismo, effettuando una sperimentazione cross-dataset. Il modello viene addestrato su un dataset contente commenti sarcastici, e testato su un dataset contente commenti aggressivi, ottenendo un F-score del 88%.

Questo esperimento dimostra che esiste un legame tra il sarcasmo e i commenti contenenti cyberbullismo.

# Capitolo 3 – Progettazione

Machine learning

Il lavoro di questa sfrutta le ultime conoscenze nel campo del machine learning.

Il **machine learning**, o apprendimento automatico, è un modalità per  **l’attuazione dell’intelligenza artificiale;** può essere definito come un sottogruppo dell’AI che si concentra sulla capacità delle macchine di ricevere una serie di dati e di apprendere da soli, modificando gli algoritmi man mano che ricevono più informazioni su quello che stanno elaborando.

Lo scopo di questo capitolo è fornire una panoramica generale su queste tecnologie e sulle loro applicazioni con particolare enfasi a quelle utilizzate nel presente lavoro di tesi.

## 3.1 Introduzione al machine learning

La definizione più citata di machine learning o apprendimento automatico è quella fornita da Tom M. Mitchell: “Si dice che un programma apprende dall’esperienza E con riferimento ad alcune classi di compiti T e con misurazione della performance P, se le sue performance nel compito T, come misurato da P, migliorano con l’esperienza E”.

L’idea alla base del concetto di apprendimento automatico è quindi quella di fornire ai computer l’abilità di imparare e replicare alcune operazioni senza essere stati esplicitamente programmati, dopo aver fatto esperienza su un insieme di dati di apprendimento [29]. Tali operazioni sono in genere di tipo predittivo o decisionale e basate sui dati che si hanno a disposizione.

In questo lavoro si sfrutteranno anche le tecniche di *sentiment analysis* e di *NPL*(Natural Language Processing) dei campi di ricerca del machine learning con aspetti in comune.

L’NPL pone il focus sulla comprensione di testi scritti o espressi tramite la voce in un linguaggio quanto più simile a quello degli esseri umani, che comprende quindi sinonimi, figure retoriche, variazione nella costruzione di una frase, esclamazioni, sarcasmo etc. E’ possibile utilizzare i progressi di questa branchia dell’intelligenza artificiale ogni giorno: gli assistenti vocali come Alexa o Cortana, o i servizi di assistenza on-line tramite *chatbot* ad esempio*,* sfruttano le ultime tecniche di questo campo. Uno dei task dell’NPL è il riconoscimento delle emozioni, noto anche come *Sentiment analysis* (o *Opinion Mining*) con cui indica l’insieme delle tecniche e procedure atte allo studio e l’analisi di informazioni testuali, al fine di rilevare valutazioni, opinioni, atteggiamenti ed emozioni relative ad una certa entità (prodotto, persona, argomento, etc). Questo tipo di analisi ha evidenti ed importanti applicazioni in una moltitudine di campi, a partire dal scopi di marketing fino alla rilevazione dei cyberbullismo, grazie anche alla mole di commenti, opinioni e post presenti nei social media. Tuttavia le tecniche di *sentiment analysis* presentano diversi problemi. Questo è dovuto soprattutto alla sottile distinzione che esiste tra sentimento positivo e negativo. Inoltre non sempre le opinioni sono espresse tramite l’uso di *opinion words*, ma in molti casi entrano in gioco altri artefici linguistici come le figure retoriche.

In alcuni casi nelle opinioni o nelle conversazioni testuali, così come nella realtà, vengono utilizzate frasi contenti ironia o sarcasmo dove l’interpretazione del significato è strettamente soggettivo e relegato al contesto, al punto che anche due esseri umani potrebbero trovarsi in disaccordo sull’interpretazione.

## 3.2 I paradigmi dell’apprendimento automatico

I compiti dell’apprendimento automatico vengono solitamente classificati in tre categorie [30], o paradigmi:

* Apprendimento supervisionato: si forniscono esempi degli input e dei rispettivi output desiderati, con l’obiettivo di estrarre un pattern che associ l’input all’output corrispondente
* Apprendimento non supervisionato: gli input forniti non hanno né una struttura definita né output associati. Lo scopo del calcolatore è quindi quello di identificare dei pattern negli input al fine di riprodurli o prevederli.
* Apprendimento per rinforzo: prevedere l’interazione del calcolatore con un ambiente dinamico nel quale si cerca di raggiungere un obiettivo(per esempio il superamento di un livello in un videogioco)

Le tecniche utilizzate in questa tesi afferiscono alla branchia dell’apprendimento supervisionato.

Il processo di apprendimento di un algoritmo di *machine learning* segue diverse fasi [29]:

* Preprocessing: in questa fase si “trattano” i dati grezzi, estraendoli da un dataset e filtrandoli per renderli adatti alla fase seguente. Esempi comuni di operazioni tipiche di questa fase sono la rimozione di *stop* *word*, ovvero quelle parole molto comuni che non aggiungono significatività a un testo, o la rimozione di caratteri speciali. In generale una fase troppo aggressiva di preprocessing potrebbe portare ad eliminare delle informazioni vitali per la corretta classificazione. Come chiariranno gli esperimenti più avanti, lo *stemming* sarà un operazione di quelle.
* Feature extraction: Una feature è una propietà o caratteristica osservabile di un fenomeno. In questa tesi verranno utilizzate e messe a confronto due tipi di feature, ovvero il *tf-idf* e il *word embedding*. Entrambi verranno descritti nell’ terzo capitolo.
* Addestramento del modello: in questa fase il modello scelto apprende attraverso le feature, per elaborare un modello previsionale. In questa tesi si è deciso di mettere a confronto tre tipi di modelli *shallow learning* e tre tipi di modelli *deep learning*, sfruttando le migliori combinazioni di preprocessing e feature extraction apprese dallo stato dell’arte.
* Test e valutazione: Per valutare la bontà di un modello vengono utilizzate metriche come precisione, richiamo, F1-score e accuratezza, che verranno esplicitate nel quarto capitolo. Inoltre verranno utilizzate tecniche di *cross* *validation,* [31]delle tecniche statistiche utilizzate per evitare problemi di sovradattamento e di campionamento asimmetrico.

Tipicamente si possono distinguere due tipi di approcci: *shallow* e *deep learning*.

## 3.3 Shallow learning

I modelli *shallow learning* si differenziano dai modelli *deep learning* principalmente per il modo in cui apprendono.

Nei modelli *shallow* la fase di feature extraction è un processo manuale che richiede la conoscenza del dominio dei dati che vogliamo classificare. Nella tesi verranno utilizzati tre modelli di questo tipo: Random Forest, Support Vector Machine, Logistic Regression , dato che come evidenziano gli studi allo stato dell’arte [15] [16], si sono dimostrati particolarmente solidi.

### 3.3.1 Random Forest

Rappresenta un tipo di modello ensemble, che si avvale del *bagging* come **metodo di ensemble** e l’albero decisionale come **modello individuale** [32]**. Ciò significa che combina molti alberi decisionali in un unico modello. Individualmente le previsioni fatte dagli alberi decisionali potrebbero non essere accurate, ma combinate insieme le previsioni saranno in media più accurate. Questo meccanismo viene chiamato “voto di maggioranza”.**

**Il risultato finale in un problema di classificazione è la classe restituita dal maggior numero di alberi. (figura 8)**

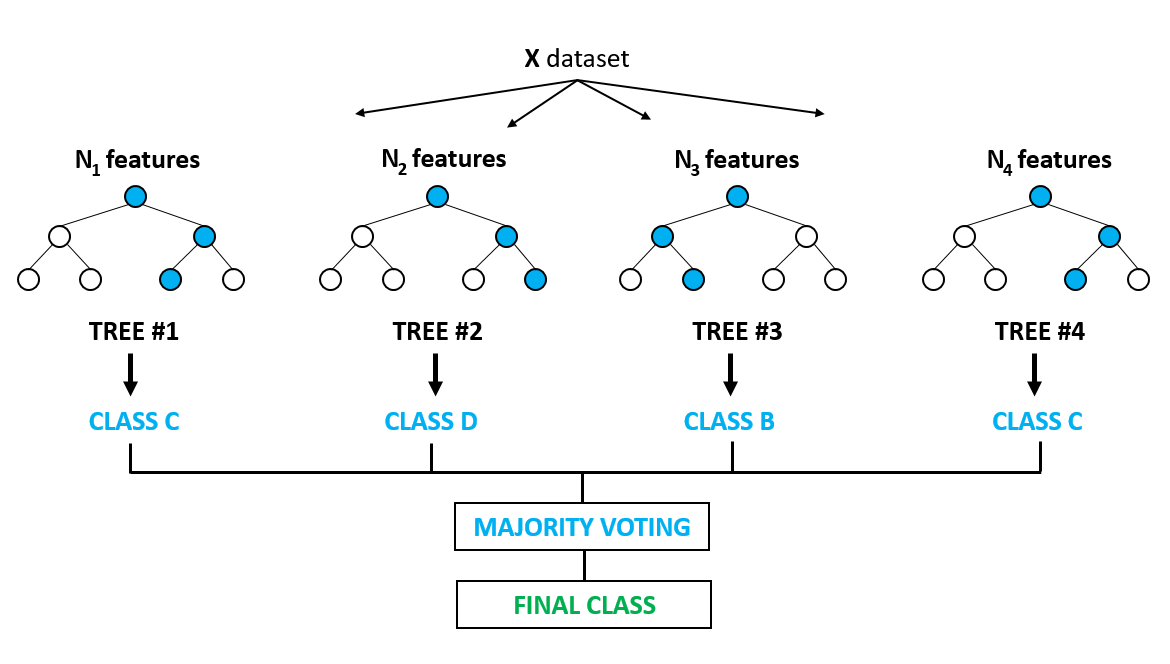


Figura 8. Random Forest

### 3.3.2 SVM

Questo modelli si basa sull’idea di trovare un iperpiano che divida al meglio il set di dati in due classi [33].

Per fare ciò utilizza i *support* *vector,* ovvero i punti dati più vicini all’iperpiano. Tali punti dipendono dal set di dati che si sta analizzando e se vengono rimossi o modificati alterano la posizione dell’iperpiano divisorio.

L’iperpiano, o la retta nel caso si stia lavorando a due dimensioni, viene tracciato a metà del margine, ovvero a metà della distanza tra i vettori i vettori di supporto di due classi differenti. Se tale iperpiano non esiste, SVM utilizza una mappatura non lineare per trasformare i dati di allenamento in una dimensione superiore. In questo modo i dati di due classi possono sempre essere separati da un iperpiano, che sarà scelto per la suddivisione dei dati(figura 9). La SVM è uno dei classificatori più precisi, tuttavia come si vedrà negli esperimenti più avanti mostra delle gravi carenze in caso di dataset non bilanciato.

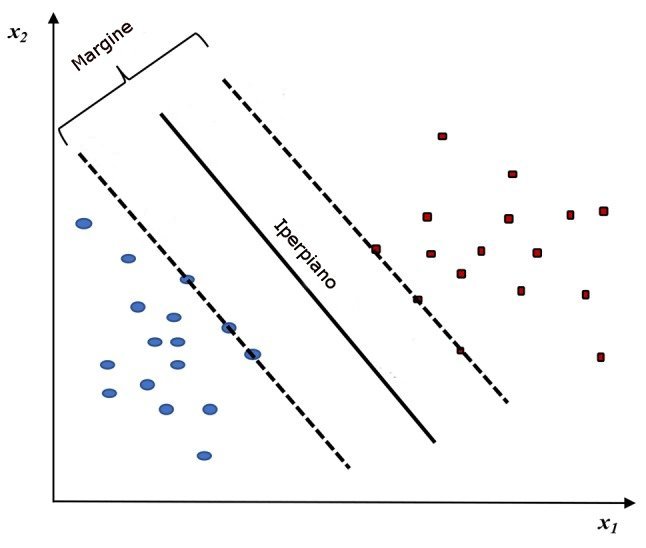


Figura 9. Rappresentazione di un iperpiano che divide gli esempi del training

### 3.3.3 Logistic Regression

Un modello di regressione logistica calcola la somma pesata delle *feature,* con l’aggiunta di un termine denominato *bias*. [29] Il risultato verrà processato da una funzione logica chiamata *sigmoid,* una particolare funzione con un andamento a ‘S’ esprimibile con la seguente equazione:

L’analisi visiva di questa funzione viene riportata nella figura 10. In questo modo dato qualsiasi input, l’output sarà sempre compreso fra due valori(0 e 1), permettendo quindi una classificazione binaria.

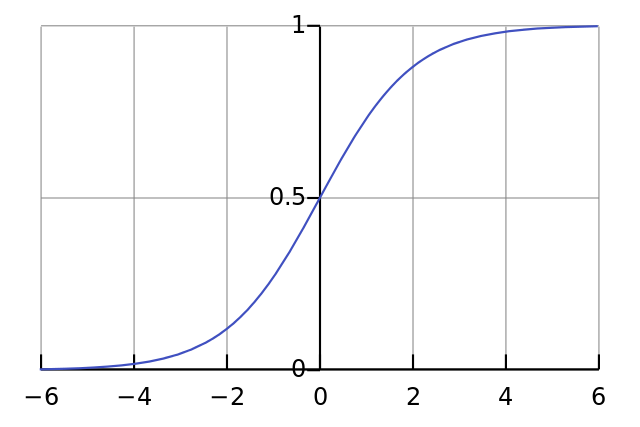


Figura 10

## 3.4 Deep Learning

I modelli *deep* *learning,* a differenza dei modelli *shallow,* apprendono in maniera automatica: essi analizzano in modo autonomo le feature o i dati grezzi, e sono in grado di classificare un testo o un oggetto riconoscendone, autonomamente, le caratteristiche.

Seppur la richiesta di capacità computazionali enormi possa rappresentare un limite, **la scalabilità del *deep learning* grazie all’aumento dei dati disponibili e degli algoritmi è ciò che lo differenzia dai modello *shallow***: i modelli di *deep* *learning*, infatti, migliorano le proprie prestazioni all’aumentare dei dati mentre i modelli come SVM o Random Forest una volta raggiunto un certo livello di performance non sono più scalabili nemmeno aggiungendo esempi e dati di training.

Un approccio deep learning è caratterizzato dall’utilizzo di una rete neurale, che può essere di tipo ricorrente o di tipo feed-forward.

Una rete neurale artificiale (Artificial Neural Network, ANN) è un sistema di deep learning ispirato alle reti neurali biologiche. Si tratta di un modello matematico di calcolo basato su percettroni, neuroni artificiali in grado di apprendere, ovvero accumulare esperienza. Il percettrone è molto simile ai neuroni biologi, le unità fondamentali del nostro cervello ed è stato proposto nel 1958 da Frank Rosenblatt. Tuttavia la complessità computazionale richiesta per addestrare una rete di neuroni ha fatto sì che prendesse piedi solo nei decenni successivi.

In figura 11 è possibile osservare un percettrone artificiale.

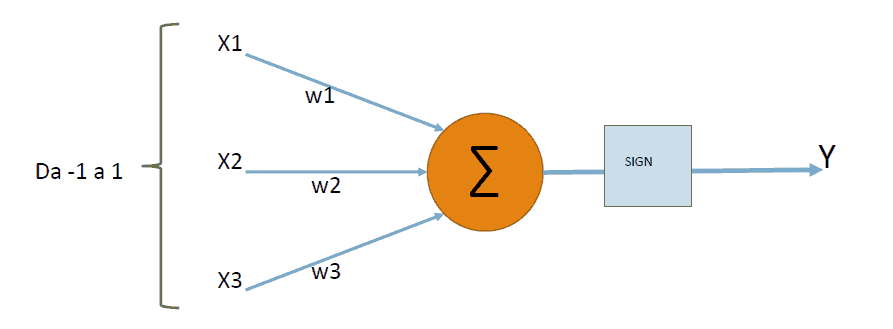


Figura 11. Un percettrone con più input di ingresso e solo un output in uscita

Una rete neurale è organizzata in strati di neuroni: essa riceve le informazioni d’ingresso attraverso un input layer e restituisce i risultati dell’elaborazione per mezzo di un output layer; tra questi strati possono esserci uno o più hidden layer(fig. 12 ).

Il numero di strati e collegamenti tra i neuroni nella rete è tale che può diventare difficile calcolare le “regolazioni” che devono essere apportate in ogni fase del processo di addestramento. Per questo si usa un **algoritmo di retro propagazione dell’errore**(**backpropagation**), proposto nel [1986](https://it.wikipedia.org/wiki/1986) da [David E. Rumelhart](https://it.wikipedia.org/w/index.php?title=David_Rumelhart&action=edit&redlink=1), G. Hinton e R. J. Williams, attraverso il quale si rivedono i pesi della rete neurale (le connessioni tra i neuroni) in caso di errori. La rete propaga all’indietro l’errore in modo che i pesi delle connessioni vengano aggiornati nel modo più appropriato.

In questa tesi vengono confrontanti tre tipi di reti neurali: CNN, BiLSTM e BiLSTM with attention, poiché utilizzate con successo negli studi di Sweta Agrawal et al [10], V. Banerjee, Juy Telavane et al. [14], Ç. İ. Acı et al. [18] e M. Mahat [22]

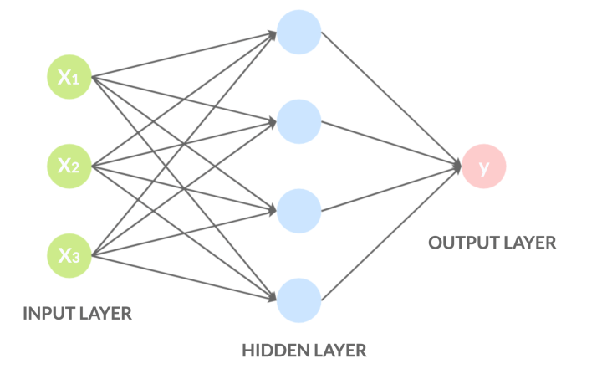


Figura 12. Una rete neurale con un solo layer intermedio

L’obiettivo di una rete neurale è quello di ridurre la differenza tra l’output previsto e l’output effettivo. Questa differenza può essere rappresentata da una funzione di costo, la cui minimizzazione attraverso l’ottimizzazione dei pesi porta a una maggior accuratezza.

Gli ottimizzatori hanno quindi il compito di modificare gli attributi di una rete neurale, come pesi e velocità di apprendimento, e sono tutti basati sulla discesa di gradiente. Di seguito alcuni esempi:

* NAG(*Nesterov Accelerate Gradient*). La discesa di gradiente prende in considerazione fattori come il “momento” e lo “slancio”. In questo modo la discesa subirà un accelerazione quando possibile, ma si arresterà se subisce una variazione anormale. [34]
* Adam(*Adaptive* *Moment* *Estimation*). Ottimizzatore che utilizza un tasso di apprendimento adattivo. Utilizza il calcolo della media del gradiente, insieme alla varianza. [35]
* Nadam. Metodo ibrido che unisce NAG e Adam. Il processo di apprendimento viene accelerato sommando il decadimento esponenziale delle medie mobili per il gradiente precedente e corrente. Inoltre, permette di compiere uno step più accurato nella direzione del gradiente aggiornando i parametri con il momento prima di calcolare il gradiente [36]

Nelle reti neurali implementate si useranno Adam e Nadam.

### 3.4.1 Reti Feed-Forward

Le reti *feed-forward* sono reti neurali dove le connessioni tra le unità non formano cicli, differenziandosi dalle reti neurali ricorrenti [37]

Nella prima sperimentazione verrà impiegata una rete neurale convoluzionale (CNN o ConvNet), una rete aciclica di neuroni artificiali il cui il modello di connessione tra i neuroni è ispirato alla corteccia visiva animale [38]. I neuroni di questa regione del cervello sono disposti in modo tale da corrispondere a regioni sovrapposte del campo visivo.

Le ConvNet consistono di una pila multistrato di percettroni, il cui scopo è quello di processare piccole quantità di informazione. Le reti neurali convoluzionali si stanno affermando come stato dell’arte per una vasta gamma di applicazioni nei campi della computer vision e del *natural language processing*. Nella classificazione di un testo gli inputs della rete neurale sono frasi o documenti rappresentati sotto forma di matrice dove ogni riga della matrice corrisponde a una parola.

Tipicamente è composta da due tipi di neuroni:

* I neuroni di elaborazione, che elaborano una porzione limitata del testo attraverso una funzione di convoluzione
* I neuroni di aggregamento o di pooling.

L’operazione di convoluzione ha il compito di estrarre le *feature* dai dati di input. Essa viene applicata per mezzo di una *sliding* *windows,* unapiccola matrice i cui valori caratterizzano l’operatore, che viene fatta traslare sulla matrice di input, eseguendo il prodotto scalare dei valori [39].

La matrice contiene i pesi utilizzati per calcolare la score function e sono associati a un bias che viene sommato al risultato della convoluzione.(fig. 13)

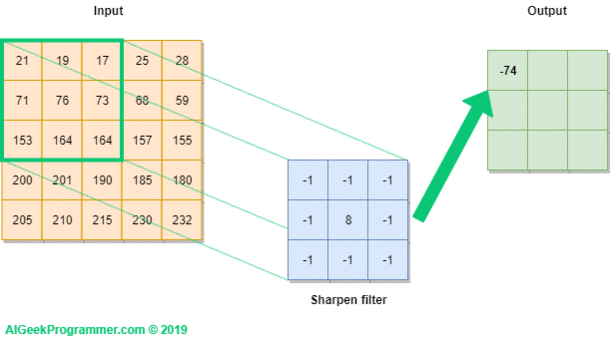


Figura 13

Dopo un layer convoluzionale si utilizza un layer di pooling. I filtri del layer convoluzionale forniscono informazioni pure, perché evidenziano alcune caratteristiche eliminandone altre che costituirebbero rumore, rendendo inutile portarsi tutta l’informazione dietro. In questo modo viene risparmiata memoria, che implica un minor costo computazionale. Il layer di pooling serve quindi a ridurre la dimensione delle informazioni elaborate, ed avviene attraverso il meccanismo della sliding windows analogo all’operazione di convoluzione. La CNN implementata per gli esperimenti di questa tesi utilizza un tecnica di pooling globale, ovvero ad ogni spostamento del kernel verrà restituito soltanto il valore maggiore(fig. 14).

Immagine che contiene testo, orologio

Descrizione generata automaticamente

Figura 14. La matrice di destra è ottenuta prendendo

i valori maggiori dalle sottomatrici a sinistra

### 3.4.2 Reti Ricorrenti

Le reti ricorrenti(*Recurrent Neural Network, RNN*) prevedono, oltre alle connessioni con i livelli successivi, anche delle connessioni con i livelli precedenti che vanno a formare dei *loop* [40], che rappresentano il concetto di “memoria” di una rete. Di conseguenza in una RNN l’output di un neurone può influenzare se stesso o può influenzare i neuroni della catena precedente.

Una rete ricorrente trova quindi la sua implementazione ideale nei problemi di *sentiment* *analysis* o NPL(*Natural* *Language* *Processing*), ma anche nell’elaborazione di video o di qualsiasi segnale temporale. Possono essere impiegati per la predizioni economiche dei titoli azionari, o per il rilevamento di anomalie nel traffico o in molti altri ambiti. Nel riconoscimento di un testo, i dati di input saranno le singole parole e nell’elaborazione si terra conto anche della parola precedente, in modo da inquadrare il giusto contesto della frase. Per poter essere addestrate le reti ricorrenti utilizzano il concetto di *cella*, una parte di rete ricorrente che preserva uno stato (o memoria) interno h(t) per ogni istante temporale, costituita da un numero prefissato di neuroni. H(t) dipende quindi dall’input x(t) e dallo stato precedente h(t-1)

In questo lavoro di tesi si useranno due tipi di reti ricorrenti: BiLSTM e BiLSTM with attention, entrambe sono basate sulle reti LSTM.

Le LSTM sono un tipo particolare di rete neurale ricorrente, che ha il vantaggio di apprendere da lunghe sequenze temporali e conservarne la memoria; particolarità che le rende ideali nel campo dell’NPL e della *sentiment analysis*, ma anche nell’analisi di sequenze di video. L’idea alla base dell’architettura LSTM è quella che ad ogni time step alcuni gate sono utilizzati per controllare il passaggio di informazioni lungo le sequenze che sono in grado di catturare dipendenze sul lungo termine più accuratamente.

In una cella LSTM lo stato h(t) viene suddiviso in due vettori componenti: h(t) e c(t). Il primo vettore è quello responsabile della memoria a breve termine, il secondo invece si occupa della memoria a lungo termine. Ciò che deve essere immagazzinato in questi tipi di memoria viene deciso dai gate.(fig. 15)

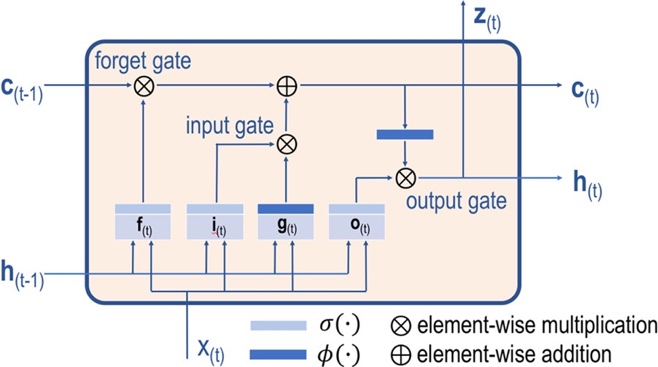


Figura 15. Una cella LSTM

Il flusso di informazioni all’interno di una cella viene gestito da tre tipi di gate: input, output, e *forget*.

* Nella prima fase il *forget* *gate* considera lo stato precedente e il nuovo input, decidendo quali informazioni mantenere e quali dimenticare. Applicando la funzione sigmoide, i valori saranno compresi tra il valore 0 e il valore 1, dove lo zero sta ad indicare l’eliminazione dell’informazione.
* Il passaggio successivo avviene tramite *l’input* *gate*, che ha il compito di determinare quali informazioni saranno aggiunte nello stato a lungo termine della cella. Verrà applicata una funzione di attivazione di tipo *tanh* per avere un output di valori compreso fra [-1,1], e successivamente sigmoide.
* L’ultimo passaggio temporale è dato *dall’output* *gate*. Questo gate, come suggerisce il nome, si occupa di decidere quale sarà il nuovo stato nascosto, ovvero la nuova memoria. In questo stato si moltiplicheranno stato precedente e input con una funzione sigmoide in primis e *tanh* in seguito. L’output di questa operazione rappresenterà il nuovo stato nascosto.

L’addestramento in una LSTM avviene in maniera unidirezionale, ovvero i dati verranno elaborati da sinistra verso destra all’interno della cella. Di conseguenza essa ricorderà le “dipendenze” in quest’ultima direzione.

Una rete ricorrente Bi-LSTM è il risultato della combinazione di due LSTM. L’idea è quella di addestrare la rete in maniera bidirezionale, in modo che la cella oltre a ricevere i dati da sinistra verso destra, li riceva anche da destra verso sinistra. In questo modo si utilizzeranno le informazioni provenienti dal passato e dal futuro [41], per decifrare il giusto contento di una frase. In figura 16 è possibile notare le differenze tra una LSTM e una BiLSTM generica.

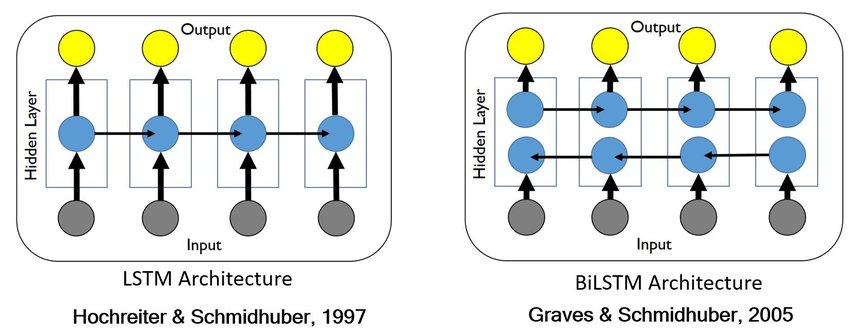


Figura 16. LSTM e BiLSTM a confronto.

Come si vedrà più avanti nel capitolo dello stato dell’arte, questa architettura raggiunge un’ottima performance nel riconoscimento del sarcasmo [27]

L’ultima architettura deep proposta è una BiLSTM che sfrutta un meccanismo di attenzione [42].

L’idea di attenzione deriva dall’osservazione che ogni parola contribuisce al significato della frase in maniera diversa; non tutte le parole infatti contribuiscono con lo stesso “peso”. Un utente qualsiasi(o vittima) coglie in maniera intuitiva le parole più significanti di una frase. Questo meccanismo viene implementato attraverso un layer di attenzione, reso disponibile dalla libreria *Keras*. In figura 17 un esempio di BiLSTM con layer di attenzione e word embedding.

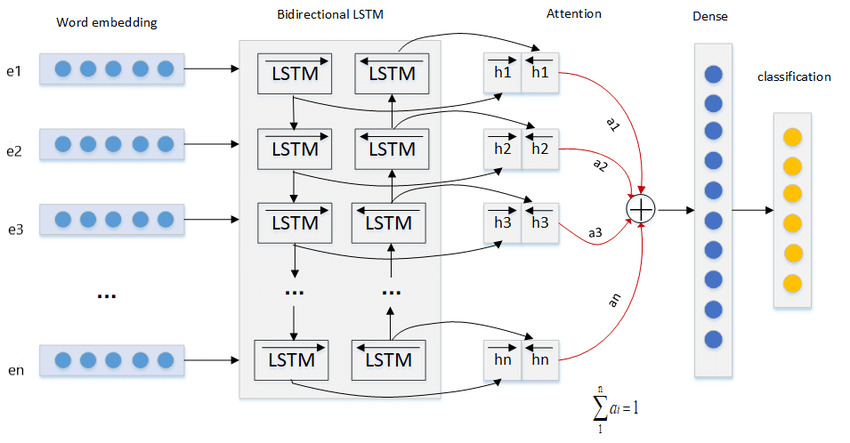


Figura 17

## 3.5 Dataset

Le sperimentazioni si articolano su 5 dataset:

* FormSprings: dataset contenente 12773 istanze, di cui 776 contenenti cyberbullismo. Esso è stato costruito attraverso la tecnica dello *scraping,* ed è stato etichettato manualmente attraverso la piattaforma di crowdsourcing Amazon Mechanical Turk [43]. Per il suo utilizzo è stato necessario rimuovere tutti i valori NaN al suo interno.
* Cyber-Troll: composto da 20.000 Tweets, di cui 7822 contenenti cyber-aggressioni, etichettato manualmente.[[4]](#footnote-4)
* Tweets: dataset contenente 11.407 tweets, etichettati con sentimento positivo(2292) e negativo(9115). E’ stato creato effettuando uno *scraping* sui profili di Twitter delle più grandi compagnie aeree. Per il suo corretto funzionamento si è reso necessario una lavorazione di base, che consiste nell’eliminazione dei tag quali “@AmericanAir”, “@USAirways” etc. e nella rimozione dei tweets da sentimento neutro.
* Large Movie Review Dataset: dataset contenente 25.000 recensioni di film, divise equamente in positive e negative.[[5]](#footnote-5) Unico dataset bilanciato dello studio. Le recensioni vengono fornite in documenti testuali, perciò è stato necessario creare un *Dataframe* agglomerando le varie recensioni e aggiungendo la colonna per la descrizione del sentimento.
* Fracking Sarcasm: dataset creato da Aniruddha Ghosh et al. [28] con la tecnica dello *scraping,* cercando tweets con l’hastag #sarcasm ed estendendo la lista dei possibili hastag con un approccio LSA. Questo dataset verrà impiegato nell’ultimo esperimento.

## 3.6 Metriche utilizzate

Per valutare la bontà dei classificatori con i vari dataset verranno utilizzate metriche tipiche del *Machine Learning* e *Information Retrieval,* quali *precision*, *recall*, e F1-score.

La *precision* viene utilizzata per misurare i modelli positivi(*true positive*) che sono **classificati correttamente** dai modelli totali(*true* *positive* più *false positive*) previsti in una **classe positiva**. [44] Essa va sempre utilizzata insieme ad altre metriche, poiché in caso di nessuna previsione falsa positiva (*FP*) si otterrebbe uno *score* del 100%, valore che potrebbe non rispettare le reali performance del classificatore.

La recall invece indica il rapporto di istanze positive correttamente individuate dal classificatore o rete neurale.

L’F1-score è la media armonica di *precision* e *recall*. Rispetto ad una media convenzionale, quella armonica attribuisce un peso maggiore ai valori piccoli in modo che un classificatore ottenga un alto punteggio solo quando *precision* e *recall* sono entrambi alti. Per questo motivo rappresenta una delle metriche più veritiere sulle reali capacità di un classificatore.

## 3.7 Librerie utilizzate

Per realizzare i vari esperimenti si utilizzeranno le seguenti librerie:

Keras: libreria open-source per l’implementazione delle reti neurali, scritta in Python. Permette di costruire reti profonde aggiungendo o modificando “layer”, le unità fondamentali dell’apprendimento profondo. Supporta inoltre vari tipi di ottimizzatori e motori di calcolo back-end come Tensorflow. Le funzioni utilizzate saranno:

* *Tokenizer:* Effettua la tokenizazzione delle frasi. Questa classe verrà utilizzata nei modelli deep learning.
* *fit\_on\_texts:* metodo per la vettorizzazione del testo.
* *texts\_to\_sequences:* metodo per trasformare ogni sequenza in una lista di testo.
* *pad\_sequences:* trasforma una lista in un 2D Numpy array nella forma (n. istante, n. timesteps)

Nei modelli deep learning verranno utilizzati altri due metodi per ottenere una cross-validation:

* *KerasClassifier*: classe che ci permette di richiamare una rete neurale più volte per scopi di valutazione.
* *StratifiedKFold*: metodo che effettua una fold cross validation.

Tensorflow: libreria open-source per l’apprendimento automatico. Sviluppato da Google Brain è alla base di molti prodotti Google come Gmail, riconoscimento vocale etc. Supporta numerose CPU e GPU per il calcolo multithread.

NTLK: libreria per l’elaborazione del linguaggio naturale. Verrà utilizzata per le varie operazioni di preprocessing, quali rimozione di *stopword* o tokenizzazione. Sono state utilizzate le seguenti funzioni:

* *Word\_tokenize:* classe che permette la tokenizzazione;
* *PorterStemmer:* classe che implementa il Porter stemming;
* *TfidfVectorizer:* classe che consente di estrarre i pesi tf-idf.

Pandas: libreria open source per l’analisi e la manipolazione dei dati. Verrà utilizzata per importare ed analizzare i dataset. In particolare:

* *Read\_csv* e *read\_json* per effettuare l’importazione e la manipolazione dei dataset.

Scikit-learn: libreria principe per il test e lo sviluppo di modelli di machine learning. Permette l’addestramento, il testing e la valutazione di un modello, anche con cross-validation.

* *RandomForest*, *Svm, LogistiRegression*: classi che istanziano gli omonimi classificatori;
* *TfidfVectorizer*: classe che permette di estrarre i pesi tf-idf

Gensim: libreria che offre una serie di utility per le operazioni di NPL. In particolare verrà utilizzata:

* *KeyedVectors* : classe che ci consente di importare i file per il GLOVE embedding;
* *get\_vectors*: che ritorna i vettori di valori reale delle parole all’interno dei dataset.

## 3.8 Primo esperimento

Per trovare le migliori combinazioni di preprocessing e feature extraction nel primo esperimento si è deciso di testare le migliori soluzioni fornite dallo stato dell’arte, in particolare dagli studi:

* Sweta Agrawal et al. [10] che hanno utilizzato word embedding sulle piattaforme di Twitter, FormSprigs oversamplato e Wikipedia;
* S.Sadiq et al. [19] che hanno utilizzato un approccio basato sul tf-idf tramite SVM;
* S. Muhamad, L. Ashanti et al. [20] che propongono l’uso di l’applicazione di n-grammi insieme al tf-idf;
* V. Banerjee, Juy Telavane et al. [14] che tramite l’utilizzo di GLOVE embedding e di una CNN raggiungono un accuratezza del 94%.

In particolare verranno sperimentate le varie combinazioni:

* Standard preprocessing con GLOVE embedding;
* Standard preprocessing con stemming e GLOVE embedding;
* Standard preprocessing con TF-IDF in termini di *unigram* e *bigram*.

Con standard preprocessing si intendono le seguenti operazioni:

* Rimozione del rumore, ad esempio numeri, spazi, punteggiatura, hastag etc.
* Rimozione di *stop word;*
* Tokenizzazione, ovvero la suddivisione del testo in *token,* dove ogni unità rappresenta una parola.

Il GLOVE embedding e il tf-idf sono due modi differenti di feature extraction:

Il tf-idf è una funzione utilizzata nell’ambito dell’Information Retrieval per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti [45]. Essa aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento, ma cresce in maniera inversamente proporzionale con la frequenza del termine nella collezione. L'idea alla base di questo comportamento è di dare più importanza ai termini che compaiono nel documento, ma che in generale sono poco frequenti.

E’ composto da due parti:

* Tf (*Term* *Frequency*), frequenza della parola in una frase;
* Idf(*Inverse* *Document* *Frequency*), frequenza inversa di una parola in una frase. Più il termine sarà specifico e più sarà alto il suo Idf.

La sua formula viene riassunta nella figura 18.

Immagine che contiene testo

Descrizione generata automaticamente

Figura 18

Ad esempio, supponendo che in un generico tweet di 50 termini la parola “idiota” compaia 5 volte, il suo tf si calcola come 5/50 = 0.1 Assumiamo ora di avere 500 tweets, e che la stessa parola compaia in 10 di questi tweets. Il suo Idf verrà calcolato come logaritmo in base 10 di 500/10 = 1.7. Il suo tf-idf si calcolerà quindi come 0.1 x 1.7 = 0.17.

In termini di *unigram* e *bigram* significa che la frase verrà “spezzata“ in parole singole e parole doppie. Ciò significa che nel caso di un’ipotetica frase “Il bambino sta giocando in strada”, si avrà in termini di *bigram*: “il bambino”; “sta giocando”; “in strada”, mentre in termini di *unigram* “il” ,”bambino”, “sta”, “giocando”, “in”, “strada”. In particolare si sono scelti i termini *unigram* e *bigram* poiché utilizzata da S. Sadiq, A. Mehmood et al. [19] con ottime performance.

Il word embedding invece è una tecnica utilizzata per la rappresentazione delle parole nel campo della NPL. Le parole vengono codificate come vettori di valori reali in uno spazio multidimensionale, dove la somiglianza tra le parole in termini di significato si traduce in vicinanza nel vettore spazio [46]

In questo modo è possibile memorizzare le informazioni sintattiche e semantiche delle parole, elaborandone il giusto contesto in una frase. Ad esempio, dando in input ad un algoritmo di word embedding la parole “cane”, esso ci restituirà delle parole semanticamente simili, in base alla distanza espressa in coseno. Ovvero ritornerà come parola semanticamente più simile “gatto” con un valore di somiglianza del 90%, seguito da “coniglio” con una somiglianza del 70% etc.

Attualmente vengono utilizzati principalmente due tipi di algoritmi: Word2Vec e GLOVE embedding.

Word2Vec è una metodologia che consiste nella composizione di due algoritmi:

* *Continuos Bag of Word* (CBOW);
* *Skip-gram.*

Il modello CBOW consiste nel predire una parola corrente dato il suo contesto, ovvero date n parole precedenti e n parole successive. Il problema di questo approccio è che una parola può essere presente in molteplici contesti, pertanto occorre utilizzare delle tecniche che diano delle priorità ad un contesto piuttosto che a un altro.

L’architettura *skip-gram* invece utilizza un approccio opposto al CBOW, ovvero ogni iterazione cercherà di predire le n parole precedenti e le n parole successive data una parola target.

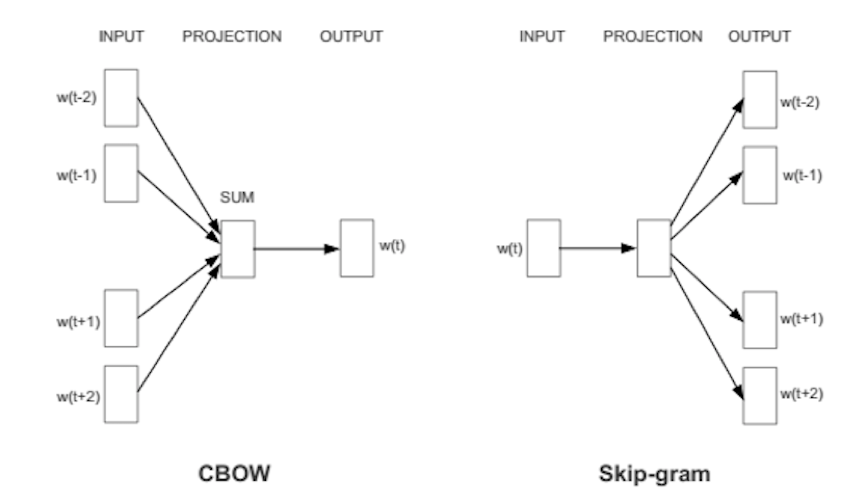


Figura 19. CBOW e Skip-gram a confronto

GLOVE è un algoritmo di apprendimento non supervisionato, creato da Jeffrey Pennington et al [47]. Esso si basa su due idee fondamentali:

* Rappresentare ogni parola tramite un vettore che riesca a catturarne il significato;
* L’utilizzo di una statistica globale in modo da non dover utilizzare solo statistiche locali per decifrare un contesto.

Esso viene addestrato su dei valori non-zero di una matrice di co-occorrenza, che rappresenta la frequenza di co-occorrenza di una parola con un'altra su un determinato dataset.

Per poter utilizzare il word embedding è necessario costruire una matrice di embedding. Essa verrà costruita attraverso il metodo *get\_vector* della libreria Gensim, applicandolo a tutti i token.

Nella seconda combinazione si sperimenterà l’influenza dello stemming nel processo di preprocessing, utilizzato con successo da V. Banerjee et al. [14]. Lo stemming è il processo di riduzione della forma [flessa](https://it.wikipedia.org/wiki/Flessione_(linguistica)) di una [parola](https://it.wikipedia.org/wiki/Parola) alla sua forma radice, detta "[tema](https://it.wikipedia.org/wiki/Tema_(morfologia))", e viene solitamente impiegato nel campo dell*’Information* *Retrieval.* Il vantaggio è di avere una versione del testo con la stessa quantità di termini, ma con meno varianti. Tuttavia presenta due criticità:

* Si può ottenere la stessa radice ma con significati differenti: ad esempio, applicando lo stemming sulle parole “biscotto” e “biscia” otterremo lo stesso tema “bisc”, anche se denota due significati molto diversi tra loro.
* Il problema opposto, ovvero stesso significato ma radice diversa: avendo “biscia” e “serpente” otterremo due temi differenti che denotano lo stesso significato.

Nelle ricerche allo stato dell’arte non è chiaro se lo stemming migliori o peggiori le prestazioni di un classificatore, per questo motivo si è deciso di testare tutti gli algoritmi con o senza stemming.

Tutte le combinazioni verranno testate su tre algoritmi di *shallow* *learning* (Random Forest, SVM, Logistic Regression), e tre algoritmi di apprendimento profondo (CNN, BiLSTM e BiLSTM con attenzione). Di seguito verranno descritte le architetture utilizzate, esplicitando i dettagli implementativi.

La rete convoluzionale implementata nel corso delle sperimentazioni sarà la seguente:

Immagine che contiene testo

Descrizione generata automaticamente

Basata sul modello Sequential, il primo livello è il layer di embedding. Rappresenta il layer di ingresso della rete neurale, e necessita di:

* Grandezza della seconda dimensione della matrice in input: il testo è stato ottimizzato tramite la funzione *pad\_sequences* con 50 valori, per tutti i dataset. Corrisponde al parametro “inp\_dim”
* Vocab\_size: numero totale di termini contenuti nel dataset.
* Embed\_size: Dimensione dei vettori pre-addestrati. Dato che Sweta Agrawal et al. [10] non rilevano particolari differenze tra 50 e 300, si è deciso di utilizzare la dimensione 50 per tutte le architetture.

Con “trainable” = True i pesi vengono aggiornati durante l’addestramento.

Il secondo layer è responsabile dell’operazione di convoluzione. Dato che l’obiettivo è la classificazione del testo, si userà *Conv1d* che opera su una sola dimensione. Per utilizzarla è necessario specificare il numero di *filters*, ovvero di percettroni, la dimensione della matrice del kernel, e la funzione di attivazione.

Il terzo layer è responsabile dell’operazione di pooling. Con pool\_size si definisce la dimensione della sliding windows.

Il quarto layer, implementato da *Flatten()* ha il compito di convertire i dati in un array monodimensionale per passarlo al layer successivo.

Gli ultimi due layer *Dense* hanno il compito di effettuare l’operazione:

output = activation(dot(input, kernel) + bias)

Il primo layer Dense sarà composto da 10 neuroni e dalla funzione di attivazione “ReLu”; dato che l’obiettivo finale è una classificazione binaria, il secondo layer sarà composto da un solo neurone, e si passerà alla funzione di attivazione *sigmoid,* che mappa qualunque valore reale in ingresso con 0 o 1 in output.

Applicando il metodo *Compile()* prepariamo il modello per l’addestramento, andando a definire la funzione di perdita(*binary\_crossentropy* dato che effettuiamo una classificazione binaria), un ottimizzatore e una metrica per la valutazione.

Come ottimizzatore è stato scelto Adam e come valutazione *accuracy;* tuttavia nel corso dell’esperimento verranno ricavate le misure quali F1-score, precisione e recall.

Queste misure verranno trovate effettuando delle predizioni e utilizzando la funzione *classification\_report* della libreria *Sklearn*. In questo modo è possibile studiare esattamente i valori di precion, recall e f1-score per ogni classe, insieme all’accuratezza globale.

La seconda rete neurale profonda sarà rappresentata da una BiLSTM:

Immagine che contiene testo

Descrizione generata automaticamente

Il layer di embedding viene implementato con la stessa logica della CNN vista precedentemente, ma subito dopo viene inserito una layer di Dropout, che ha il compito di implementare una tecnica di regolarizzazione per evitare il fenomeno dell’*overfitting*.

Con *overfitting* si intende un problema molto ricorrente nelle reti neurali, che avviene quando il classificatore si basa su delle regolarità apparenti per predire i risultati. La rete neurali quindi imparerà delle caratteristiche troppo specifiche di un training set, che non hanno riscontro nella fase di test. Alcune motivazioni comuni possono essere:

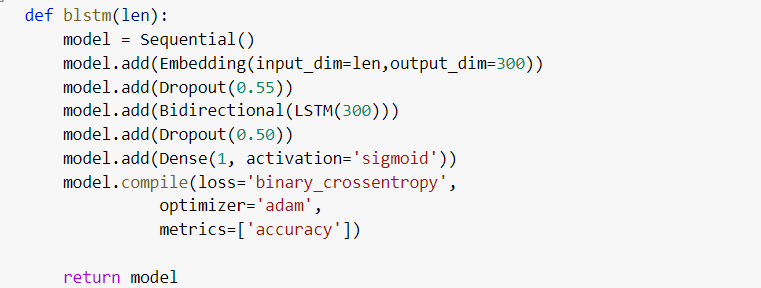
* Dati di training molto sbilanciati;
* Apprendimento troppo lungo;
* Numero di neuroni troppo alti;
* Valori dei parametri non adatti alla rete neurale.

Il layer Dense quindi spegne casualmente alcuni neuroni, evitando quindi un aggiornamento dei pesi. Il numero di neuroni da disattivare viene esplicitato dal suo unico parametro, che può variare da 0 a 1.

Successivamente il cuore della rete neurale, ovvero il layer *Bidirectional,* che andrà ad implementare la logica descritta nel paragrafo 3.4.2. Esso richiamerà al suo interno una LSTM con numero di neuroni pari al valore *embed\_size.*

La BiLSTM verrà utilizzata in questa configurazione nella prima e seconda combinazione del primo esperimento. Per utilizzarla con il tf-idf sarà necessario apportare delle modifiche ai layer di input, dato che non si utilizzerà più la matrice di embedding.

La nuova configurazione con tf-idf sarà la seguente:



Nel layer id input il parametro *input\_dim* specifica il numero di termini del dataset, mentre il parametro *output\_dim* specifica la dimensione in uscita. Essa è speculare con i numeri di neuroni utilizzati dal layer BiLSTM.

La configurazione della BilSTM con layer di attenzione sarà la seguente:

Immagine che contiene testo

Descrizione generata automaticamente

Il funzionamento del layer Embedding è analogo a quello utilizzato nelle reti neurali precedenti. Mentre successivi tre layer, *Conv1D*, *MaxPooling1D* e *Dropout* sono gli stessi descritti precedentemente.

In questa configurazione verrà sostituito l’ottimizzatore Adam con l’ottimizzatore Nadam, in quanto essere più adatto nell’addestramento di reti più complesse.

Successivamente viene creato un layer che implementa il “cuore” della BiLSTM. Questa volta il parametro *return\_sequences* verrà settato a True. Il motivo è rendere conforme l’output ritornato da questo livello per il layer di attenzione successivo. Esso infatti prende un input con tre dimensioni, mentre il layer LSTM ritorna un input con sole due dimensioni. Settando il return\_sequences a True imporremo al layer LSTM di ritornare un output su tre dimensioni.

La sua versione con il tf-idf sarà la seguente:

Immagine che contiene testo

Descrizione generata automaticamente

Verranno rimossi i layer ,*Conv1D*, *MaxPooling1D* e *Dropout* e verrà modificato il layer di Embedding.

## 3.9 Secondo esperimento

Il secondo esperimento ha l’obiettivo di comprendere se, utilizzando la miglior combinazione di preprocessing e feature extraction trovata nel primo esperimento, si riesca a migliorare la performance nel rilevamento di cyberbullismo e commenti negativi, addestrando il sistema su dati semanticamente simili.

Dato che le aggressioni tipiche del cyberbullismo possono essere considerate come un “sottoinsieme” delle frasi contenenti sentimento negativo, si addestrerà il sistema su un dataset contente sentimenti negativi e si effettuerà il test su un dataset contenente cyberbullismo, effettuando quindi una sperimentazione cross-dataset.

Il dataset da testare verrà partizionato con la funzione *train\_test\_split* utilizzando una divisione standard di 80/20. Nel dataset da addestrare ne verrà riservato il 20% per la validazione, necessaria per l’addestramento. Come metrica invece si utilizzerà il valore di f1-score nella contenente delle cyber-aggressioni.

## 3.10 Terzo esperimento

L’ultimo esperimento di questo lavoro consiste nel verificare una correlazione tra cyberbullismo e sarcasmo, seguendo gli studi di Chia Zheng et al. [6]

Per far ciò si utilizzerà la miglior combinazione di preprocessing e feature extraction ottenuta dal primo esperimento, addestrando il sistema sul dataset contente commenti sarcastici creato da Aniruddha Ghosh et al. [28] e testandolo sul dataset di Cyber-troll. Per evitare degli errori di classificazione, in fase preliminare saranno rimossi tutti gli hastag #sarcasm da questo dataset. Per escludere inoltre che una mancata correlazione possa dipendere dalla rete stessa si effettuerà un test preliminare per verificare il grado con cui essa riesce a riconoscere le istanze positive di sarcasmo.

# Capitolo 4 – Sperimentazioni

In questo capitolo verranno riportati i risultati dei vari esperimenti.

## 4.1 Primo esperimento

Come riportato nel paragrafo 3.7 del capitolo precedente il primo esperimento ha l’obiettivo, supportato dalle ricerche allo stato dell’arte, di trovare la miglior combinazione di preprocessing e feature extraction per l’identificazione di commenti aggressivi o dal sentimento negativo.

Ogni combinazione verrà messa a confronto con tre modelli di *shallow* *learning,* quali Random Forest, Support Vector Machine, Regressione Logistica, e tre architetture *deep*: CNN, BiLSTM e BiLSTM with attention.

La prima combinazione consiste nel testare una fase di preprocessing standard utilizzata dalla maggior parte degli studi, che consiste in:

* *Lowercasing,* ovvero la trasformazione da maiuscolo a minuscolo delle parole;
* Rimozione di numeri;
* Rimozione della punteggiatura;
* Rimozione di spazi extra;
* *Tokenizzazione*, ovvero la divisione delle parole per renderle adeguate all’utilizzo dei classificatori.

Come metodo di feature extraction verrà utilizzato GLOVE embedding, utilizzato nello studio [14] Per valutare correttamente le performance in ogni test è stata eseguita una *5-fold cross validation,* attraverso la funzione *Kfold* e *cross\_val\_predict* della libreria Sklearn.

La fase di standard preprocessing è stata implementata nel modo seguente:

Immagine che contiene testo

Descrizione generata automaticamente

Il classificatore Random Forest a seguito di diversi test è stato utilizzato con parametri di default, ovvero *n\_estimators* = 100, *max\_depth* = none e *max\_feature* = none. Per quanto riguarda la Support Vector Machine sono stati sperimentati diversi kernel. Seppur il kernel “poly” in alcune situazione è risultato il miglior, globalmente il kernel di tipo lineare ha avuto una maggior accuratezza e stabilità. Il classificatore Logistic Regression invece è stato utilizzato con i parametri *max\_iter* = 500 e *class\_weight* uguale a ‘balanced’.

Nella tabella seguente si possono notare i risultati ottenuti. In particolare viene riportata l’accuratezza totale, e il punteggio F1 relativa alla classe di interesse, ovvero la classe contente cyberbullismo nel caso di Cyber-Troll e FormSprings, e la classe contente commenti dal significato negativo, nei dataset Tweets e ACLL.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cyber-troll | Formsprings | Tweets | ACLL |
|  | Accuratezza/F1 score | Accuratezza/F1 score | Accuratezza/F1 score | Accuratezza/F1 score |
| Random Forest | **0.95/0.94** | 0.94/0.04 | 0.87/0.92 | 0.74/0.75 |
| SVM | 0.66/0.39 | 0.94/0 | 0.88/0.93 | 0.75/0.75 |
| LR | 0.64/0.57 | 0.72/0.23 | 0.84/0.89 | 0.75/0.75 |
| CNN | 0.87/0.85 | 0.93/0.38 | 0.89/0.93 | **0.82/0.82** |
| BiLSTM | 0.88/0.86 | **0.94/0.42** | **0.92/0.95** | **0.82/0.82** |
| BiLSTM with  attention | 0.83/0.80 | 0.94/0.40 | 0.91/0.94 | 0.80/0.80 |

Come si può notare dai risultati, il classificatore Random Forest e la rete neurale BiLSTM hanno avuto punteggi più alti. Tuttavia nel caso di dataset fortemente sbilanciato (Formspring) le performance dell’apprendimento profondo superano di gran lunga un approccio *shallow learning.*

Sul dataset ACLL i valori di accuratezza e f1-score coincidono, dato che indica un perfetto bilanciamento dei dati training. Tuttavia le performance di questo dataset rimangono al di sotto degli altri, indipendentemente dal metodo di feature extraction o di approccio. I risultati inoltre sono in linea con gli studi di S. Agrawal et al. [10], che ottengono utilizzando le stesse combinazioni di feature extraction e preprocessing un f1-score di 0.34 con CNN sul dataset FormSprings.

Su tre dataset la rete ricorrente BiLSTM si è dimostrata la più performante, seguita dal classificatore Random Forest. Ciò potrebbe indicare che le reti ricorrenti siano più abili nel decifrare il contesto di una frase. Nonostante le ottime performance delle reti ricorrenti, l’aggiunta del layer di attenzione va a peggiorare le prestazioni.

Nella seconda parte del primo esperimento si è sperimentato il ruolo dello stemming utilizzato da V. Banerjee et al. [14] nelle prestazioni di un classificatore. Come tipo di stemmer è stato utilizzato il Porter. I risultati sono mostrati in figura nella tabella sottostante:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cyber-troll | Formspring | Tweets | ACLL |
|  | Accuratezza/F1 score | Accuratezza/F1 score | Accuratezza/F1 score | Accuratezza/F1 score |
| Random Forest | **0.95/0.93** | 0.94/0.03 | 0.86/0.92 | 0.71/0.71 |
| SVM | 0.68/0.48 | 0.84/0.01 | **0.91/0.95** | 0.72/0.72 |
| LR | 0.64/0.57 | 0.71/0.22 | 0.91/0.94 | 0.72/0.72 |
| CNN | 0.87/0.84 | 0.89/0.40 | 0.89/0.93 | **0.81/0.81** |
| BiLSTM | 0.88/0.86 | **0.93/0.43** | **0.91/0.95** | 0.82/0.82 |
| BiLSTM with  attention | 0.80/0.85 | 0.94/0.36 | 0.90/0.94 | 0.79/0.79 |

I valori rimangono globalmente simili alla prima combinazione, tuttavia si può notare un leggero peggioramento dei valori f1-score, probabilmente dovuta al fatto che lo stemming porti al classificatore meno dettagli.

La rete ricorrente BiLSTM è stata l’unica che si è dimostrata stabile nelle performance, producendo gli stessi valori di accuratezza totale e f1-score nelle classi contenti aggressioni L’unico classificatore che è riuscito a migliorare le sue perfomance è la SVM. Considerato il suo funzionamento descritto nel paragrafo 3.3.2, e considerata la funzione dello stemming che è quella di ottenere più termini con meno varianti, probabilmente con questa combinazione essa riesce a disegnare più facilmente un iperpiano che divida gli esempi del training.

Nella terza combinazione si sperimenta un preprocessing standard, e un nuovo approccio di feature extraction, ovvero il tf-idf in termini di *unigram* e *bigram*.

I pesi tf-idf vengono ottenuti tramite la funzione *TfidfVectorizer* della libreria *Sklearn* con i parametri *ngram\_range* = (1, 2), in modo da selezionare i pesi *unigram* e *bigram*[[6]](#footnote-6)insieme. Un altro parametro fondamentale della funzione è *max\_feature* che seleziona il numero massimo di feature. Un valore troppo alto causa *overfitting* ai classificatori, impedendogli di apprendere come dovrebbero. A seguito di vari test si è utilizzato il valore 500 per i classificatori *shallow learning* e CNN, mentre per le architetture più complesse, ovvero BiLSTM e BiLSTM con attenzione si è reso necessario portarlo a 50 per evitare il fenomeno dell’*overfitting*.

I risultati sono mostrati nella tabella seguente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cyber-troll | Formspring | Tweets | ACLL |
|  | Accuratezza/F1 score | Accuratezza/F1 score | Accuratezza/F1 score | Accuratezza/F1 score |
| Random Forest | **0.88/0.86** | 0.94/0.28 | **0.89/0.93** | 0.85/0.85 |
| SVM | 0.73/0.65 | 0.95/0.30 | 0.91/0.73 | **0.86/0.86** |
| LR | 0.72/0.68 | 0.84/0.34 | 0.87/0.56 | **0.86/0.86** |
| CNN | 0.69/0.60 | **0.95/0.41** | 0.87/0.92 | 0.81 /0.81 |
| BILSTM | 0.64/0.28 | 0.94/0.29 | 0.83/0.90 | 0.50/0.50 |
| BILSTM with attention | 0.63/0.32 | 0.94/0.29 | 0.83/0.90 | 0.50/0.60 |

Con questo tipo di feature extraction i classificatori *shallow learning* hanno ottenuto risultati globalmente maggiori rispetto al *deep learning.* Tuttavia si è verificata una perdita di accuratezza rispetto alla prima combinazione.

Complessivamente, nel caso di word embedding come feature extraction i classificatori *shallow learning* hanno ottenuto ottime performance, tuttavia hanno mostrato delle gravi mancanze nell’utilizzo di dataset non bilanciati. I dataset Cyber-troll, Tweets e Formsprings trovano la loro miglior classificazione utilizzando questo tipo di feature extraction.

Il tf-idf invece si presta bene alla classificazione di dataset bilanciati e non, seppur con valori di accuratezza leggermente minori. L’unico dataset che predilige questo tipo di feature extraction è ACLL, probabilmente perchè gli esempi del training erano mediamente più lunghi. Un problema riscontrato con il tf-idf è che richiede una grande quantità di memoria RAM per essere implementata, poiché il metodo *TfidfVectorizer* ritorna delle matrici sparse che devono essere convertite in array prima di poter essere utilizzate. Di conseguenza, negli esperimenti successivi si è deciso di utilizzare la combinazione BiLSTM con standard preprocessing e word embedding, dato che ha ottenuto il punteggio più alto con un dataset sbilanciato e contente cyberbullismo (Formsprings), ma anche con dataset contenenti frasi dal significato negativo.

## 4.2 Secondo esperimento

Come chiarito dal paragrafo 3.8 del capitolo precedente, si andrà ad effettuare un addestramento cross-dataset, tra dataset contenti commenti negativi e dataset contenti aggressioni o cyberbullismo.

Verrà utilizzata la BiLSTM utilizzata in precedenza insieme al GLOVE embedding. Nell’addestramento si riserverà il 20% per la validazione, in modo da addestrare il sistema con il maggior numero possibile di esempi. Di seguito le varie combinazioni con i risultati, riportati attraverso il valore di f1-score della classe contenente cyber-aggressioni.

|  |  |
| --- | --- |
| Train/Test | Cybertroll |
| ACLL | 66% |
| Tweets | 61% |

I risultati mostrano una parziale correlazione tra sentimenti negativi e commenti contenenti aggressioni. Tuttavia questa correlazione si dimostra simile indipendente dalla fonte esaminata. Si ricorda infatti che ACLL è stato costruito con delle recensioni di film, mentre Tweets aggregando dei tweets trovati sui profili delle maggiori compagnie aeree americane.

## 4.3 Terzo esperimento

Come riportato nel paragrafo 3.9, l’ultimo esperimento di questa tesi ha l’obiettivo di verificare una correlazione tra sarcasmo e cyberbullismo. Anche in questo caso verrà utilizzata la stessa combinazioni di feature extraction e preprocessing degli esperimenti precedenti.

L’esperimento consiste in particolare nell’addestrare il sistema sul dataset di Aniruddha Ghosh et al. [28] e nel testing sul dataset di Cyber-troll. Per evitare che la rete apprenda delle caratteristiche non presenti nel test-set tutte le occorrenze dell’hastag #sarcasm sono state rimosse. Per verificare inoltre che il sistema sia effettivamente efficace nel rilevamento del sarcasmo lo si è addestrato e testato sul dataset stesso, attraverso una 5-fold cross validation.

Il risultato ottenuto dal test sullo stesso dataset (f1-score del 76% nella classe contente sarcasmo) dimostra la rete è in grado di generalizzare problemi anche diversi tra loro. Tuttavia nel test cross-dataset la rete ha ottenuto un punteggio del 50% di f1-score, dato che in caso di classificazione binaria indica che non vi è una particolare correlazione.

# Capitolo 5 – Conclusioni

In questa tesi si è cercato di contribuire al problema dell’identificazione del cyberbullismo esplorando vari fronti.

Si è partiti dalla ricerca del miglior sistema di preprocessing e feature extraction allo stato dell’arte, verificando le performance e il grado di adattabilità dei diversi classificatori a diversi tipi di feature extraction. Successivamente si è verificata una possibile correlazione tra commenti contenenti aggressioni e commenti dal sentimento negativo, per stabilire se i sistemi di classificazione possano beneficiare di sorgenti diverse ma semanticamente simili. L’ultima domanda che ci si è posti in questa tesi era scoprire se i cyber-bulli, al pari di quelli “tradizionali” usassero l’arma del sarcasmo per offendere o aggredire le vittime, perchè capire come gli aggressori attaccano è uno dei primi passi per la costruzione di sistemi di identificazioni più efficaci.

La fase di sperimentazione è stata organizzata in tre esperimenti ben definiti tra loro, ma dipendenti.

Tramite il primo esperimento si è trovato un ottimo sistema di feature extraction e preprocessing, composto da standard preprocessing e GLOVE embedding, in grado di classificare accuratamente sia i contenuti testuali contenti cyberbullismo e sia i commenti negativi all’interno dei social. Questa combinazione si è dimostrata inoltre molto valida nei dataset estremamente sbilanciati, permettendone l’utilizzo in molte situazioni reali. I migliori risultati ottenuti con questa combinazione sono:

* F1-score del 94% nella classe contente aggressioni sul dataset Cybertroll con l’utilizzo di Random Forest;
* F1- score del 42% nella classe contenente aggressioni sul dataset estremamente sbilanciato di Formsprings con l’utilizzo di una BiLSTM.
* F1-score del 95% nella classe contenente commenti dal sentimento negativo sul dataset di Tweets tramite una BiLSTM.
* F1-score del 82% nella classe contenente commenti dal sentimento negativo sul dataset di ACLL attraverso una BiLSTM.

Il secondo esperimento si è svolto attraverso un test cross-dataset. Il migliore sistema trovato nel primo esperimento è stato addestrato sui dataset di ACLL e Tweets, entrambi dataset contenenti commenti positivi e negativi molto utilizzati nell’ambito della sentiment analysis. Successivamente è stato testato sul dataset Cybertroll, contente aggressioni virtuali. Seppur i due dataset presentano delle differenze di contenuto, piattaforma e stili di scrittura, i risultati mostrano che vi è una correlazione parziale tra i dataset esaminati, e questa “somiglianza” rimane stabile tra i dataset.

L’ultimo esperimento di questa tesi si poneva l’obiettivo di investigare le somiglianze tra sarcasmo e cyberbullismo, con delle modalità analoghe a quelle del secondo esperimento. In questo caso i test hanno dimostrato che non è possibile stabilire un’associazione certa. Seppur il sarcasmo è una delle armi preferite dai bulli, sembra che i cyberbulli preferiscano degli attacchi più diretti, probabilmente motivati dall’anonimato in rete.

L’identificazione del cyberbullismo rimane ancora un campo ricco di possibili miglioramenti, tuttavia è possibile già raggiungere valori di accuratezza notevoli, sia attraverso i metodi di classificazione tradizionali e sia tramite il contributo delle reti neurali.

I social inoltre dovrebbero dare la possibilità ai ricercatori di estrapolare più feature dai profili utenti delle persone, in modo da poter studiare con esattezza le caratteristiche dei bulli negli ambienti virtuali. Solo un’esatta profilazione infatti, insieme ad un accurata classificazione potrà portare alla risoluzione di questo annoso problema.

# Riferimenti

|  |  |
| --- | --- |
| [1] | P. K. Smith, «Abusive Relationships in Cyberspace,» *Psychology Department.* |
| [2] | R. R, L. A. M, L. Greca, B. S.Laic, S. F.Chan e W. M.Hergeb, «Cyber victimization by peers: Prospective associations with adolescent social anxiety and depressive symptoms,» *Journal of Adolescence,* 2015. |
| [3] | B. Kowalski, G. Robin M., S. Gary W., L. Amber N. e M. R, «Bullying in the digital age: A critical review and meta-analysis of cyberbullying research among youth.,» *Psychological Bulletin,* 2014. |
| [4] | J.-B. Pingault e T. Schoeler, «Assessing the consequences of cyberbullying on mental health,» *Nature Human Behaviour ,* vol. 1. |
| [5] | S. Rosenthal, N. Farra e P. Nakov, «SemEval-2017 Task 4: Sentiment Analysis in Twitter». |
| [6] | Z. L. Chia, M. Ptaszynski, F. Masui, G. Leliwa e M. Wroczynski, «Machine Learning and feature engineering-based study into sarcasm,» *Information Processing and Management,* vol. 58, 2021. |
| [7] | K. Kumari, J. P. Singh, Y. K. Dwivedi e N. P. Rana, «Towards Cyberbullying-free social media in smart cities: a unified,» *Springer Nature,* 2019. |
| [8] | K. Kumari e J. P. Singh, «Identification of cyberbullying on multi-modal social media,» *Transactions on Emerging Telecommunications Technologies.* |
| [9] | A. Kumar e N. Sachedeva, «Multimodal cyberbullying detection using capsule network with dynamic routing and deep convolutional neural network,» *Springer Nature,* 2021. |
| [10] | S. Agrawal e A. Awekar, «Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms,» *Lecture Notes in Computer Science,* vol. 10772, pp. 141-153, 2018. |
| [11] | D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu e B. Qin, «Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification». |
| [12] | M. Dadvar e K. Eckert, «Cyberbullying Detection in Social Networks Using Deep Learning Based Models; A Reproducibility Study,» in *ECIR*, 2018. |
| [13] | M. Dadvar, D. Trieschnigg e F. d. jong, «Experts and Machines against Bullies: A Hybrid Approach to detect cyberbullies,» *Lecture Notes in Computer Scienze,* 2014. |
| [14] | V. Banerjee, J. Telavane, P. Gaikwad e P. Vartak, «Detection of Cyberbullying Using Deep Neural Network,» in *5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019. |
| [15] | V. S. Chavan e S. S.S, «Machine Learning Approach for Detection of Cyber-Aggressive Comments by Peers on Social Media Network». |
| [16] | K. D. Gorro, M. J. Sabellano, K. Gorro e C. Maderazo, «Classification of Cyberbullying in Facebook using Selenium and SVM,» in *International Conference on Computer and Communication System*, 2018. |
| [17] | T. Davidson, D. Warmsley, M. Macy e I. Weber, «Automated Hate Speech Detection and the Problem of Offensive Language». |
| [18] | Ç. İ. Acı, E. Çürük e E. S. Eşsiz, «AUTOMATIC DETECTION OF CYBERBULLYING IN FORMSPRING.ME,,» *Turkish Journal of Engineering (TUJE),* 2019. |
| [19] | S. Sadiq, A. Mehmoodb, S. Ullah, M. Ahmad, G. S. Choi e B.-W. On, «Aggression detection through deep neural model on Twitter,» *Future Generation Computer Systems,* vol. 114, pp. 120-129, 2021. |
| [20] | Noviantho, S. M. Isa e L. Ashianti, «Cyberbullying Classification using Text Mining,» in *1st International Conference on Informatics and Computational Sciences (ICICoS)*, 2017. |
| [21] | M. Gada, K. Damania e S. Sankhe, «Cyberbullying Detection using LSTM-CNN,» in *International Conference on Computer Communication and Informatics*, 2021. |
| [22] | M. Mahat, «Detecting Cyberbullying Across Multiple Social Media Platforms Using Deep Learning,» in *International Conference on Advance Computing and Innovative Technologies in Engineering(ICACITE)*, 2021. |
| [23] | Whittaker e M. Kowalski, «Cyberbullying via social media,» *Journal of School Violence,* 2015. |
| [24] | D. Chatzakou, N. Kourtellis, J. Blackburn, E. D. Cristofaro, G. Stringhini e A. Vakali, «Mean Birds: Detecting Aggression and Bullying on Twitter,» in *WebSci '17*, Troy, NY, 2017. |
| [25] | R. R. Dalvi, S. B. Chavan e A. Halbe, «Detecting A Twitter Cyberbullying Using Machine Learning,» in *Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2020. |
| [26] | P. Verma, N. Shukla e A. Shukla, «Techniques of Sarcasm Detection: a Review,» in *International Conference on advance computing innovative technologies in engineering*, 2021. |
| [27] | L. H. SON, A. KUMAR, S. R. SANGWAN, A. ARORA3, A. NAYYAR e M. ABDEL-BASSET, «Sarcasm Detection Using Soft Attention-Based,» *IEEEXPLORE,* 2019. |
| [28] | A. Ghosh e T. Veale, «Fracking Sarcasm using Neural Network,» in *7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA 2016)*, 2016. |
| [29] | C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006. |
| [30] | R. Stuart e N. Peter, Artificial Intelligence: A Modern Approach (2nd ed.), Prentice Hall, 1995. |
| [31] | T. Hastie, R. Tibshirani e J. Friedman, The Elements of Statistical Learning, Springer. |
| [32] | T. K. Ho, « The Random Subspace Method for Constructing Decision Forests,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, n. 8, 1998. |
| [33] | B. Scholkopf, A. J. Smola, R. C. Williamson e P. L. Bartlett, «New Support Vector Algorithms». |
| [34] | Y. Nesterov, «A method for unconstrained convex minimization problem with the rate of convergence,» *Doklady ANSSSR,* vol. 269, 1983. |
| [35] | D. P. Kingma e J. L. Ba, «ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,» in *ICLR*, 2015. |
| [36] | A. Tato e &. R. Nkambou, «IMPROVING ADAM OPTIMIZER,» in *Workshop track- ICLR*, 2018. |
| [37] | A. Zell, Simulation Neuronaler Netze [Simulation of Neural Networks], Addison-Wesley, 1994. |
| [38] | M. Matusugu, K. Mori, Y. Mitari e Y. Kaneda, « Subject independent facial expression recognition with robust face detection using a convolutional neural network,» *Neural Networks,* vol. 16. |
| [39] | G. Aurélien, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2019. |
| [40] | E. Prati, Mente artificiale, EGEA, 2017. |
| [41] | M. Schuster e K. K. Paliwal, «Bidirectional recurrent neural networks,» *IEEE Transactions on Signal Processing ,* 1997. |
| [42] | M.-T. Luong, H. Pham e C. D. Manning, «Effective Approaches to Attention-based Neural Machine Translation,» *ArXiv ,* 2015. |
| [43] | R. K, A. Kontostathis e L. Edwards, «Using Machine Learning to Detect Cyberbullying,» in *10th International Conference on Machine Learning and Applications Workshops (ICMLA 2011).*, 2011. |
| [44] | M. Hossin, «A Review on Evaluation Metrics for Data Classification Evaluations,» in *International Journal of Data Mining & Knowledge Management Process* , 2015. |
| [45] | A. .Rajaraman e J. Ullman, Mining of Massive Datasets.. |
| [46] | J. Daniel e H. J. Martin, Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition, Prentice Hall, 2000. |
| [47] | J. Pennington, R. Socher e C. D. Manning, «GloVe: Global Vectors for Word Representation». |
| [48] | C. V. HeeI, G. Jacobs, C. Emmery, B. Desmet, E. Lefever, B. Verhoeven, G. D. Pauw, W. Daelemans e V. Hoste, «Automatic detection of cyberbullying in social media text,» *PLoS ONE ,* 2018. |
| [49] | M. Fortunatusa, P. Anthonya e S. Chartersa, «Combining textual features to detect cyberbullying in social media posts,» *Procedia Computer Science ,* vol. 176, p. 612–621, 2020). |
| [50] | K. S. Alam, S. Bhowmik e P. R. K. Prosun, «Cyberbullying Detection: An Ensemble Based Machine Learning Approach,» in *Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV 2021).*, 2021. |

# Ringraziamenti

Arrivati alla fine di questo percorso desidero ringraziare tutte le persone che mi hanno accompagnato in questo viaggio.

In primis vorrei ringraziare la mia famiglia ed in particolare i miei genitori, per avermi dato l’opportunità di vivere questa esperienza.

Un ringraziamento particolare va anche a tutti i miei amici e compagni di avventura con i quali ho condiviso i momenti migliori e peggiori di questi anni. Senza di voi non sarebbe stata la stessa cosa.

Desidero inoltre ringraziare il Dott. Vincenzo Gattulli per la disponibilità e il supporto ricevuto durante la stesura di questa tesi, e il Prof. Donato Impedovo per la passione trasmessa durante le sue lezioni.

1. https://it.wikipedia.org/wiki/Ironia [↑](#footnote-ref-1)
2. https://www.oxfordlearnersdictionaries.com/definition/english/sarcasm. [↑](#footnote-ref-2)
3. https://www.kaggle.com/ [↑](#footnote-ref-3)
4. https://www.kaggle.com/dataturks/dataset-for-detection-of-cybertrolls [↑](#footnote-ref-4)
5. https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews [↑](#footnote-ref-5)
6. https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html [↑](#footnote-ref-6)