Università degli Studi di Salerno

Dipartimento di Informatica

Progetto Intelligenza Artificiale
Identificazione di Fake Review Amazon con NLP

Professore
Vincenzo Deufemia
Stefano Cirillo

Studenti
Antonio Cimino 522500922
Vincenzo De Martino 0522500966

Sommario

Abstract

Nel seguente documento è stato studiato un metodo per l'individuazione delle recensioni false che si possono abitualmente trovare nei siti per la vendita di prodotti, come ad esempio Amazon. Il processo ci ha portato alla realizzazione di un sistema che raccoglie informazioni delle recensioni dai siti, le etichetta secondo determinati parametri in false o non false, e poi, attraverso l'uso di tecniche di machine learning le analizza per imparare ad individuare ulteriori recensioni false.

1) Introduzione

Nel 2019 nasce il fenomeno delle recensioni false sui siti per la vendita di prodotti, come Amazon, in cambio di prodotti gratuiti, per aumentare la credibilità dei venditori. Le recensioni false si stanno espandendo in ogni tipologia di acquisto online, dai siti di e-commerce agli alberghi e persino per i negozi fisici che si possono trovare nelle nostre città. Questo è dovuto al fatto che sempre più persone stanno acquistando prodotti tramite internet, e prima di acquistare un determinato prodotto valutano se l'acquisto è sicuro tramite le valutazioni fatte dalle altre persone che hanno acquistato il prodotto in precedenza, non potendo valutare essi stessi il prodotto. Di conseguenza un prodotto con scarse valutazioni vende meno, e anche il venditore viene valutato in base alle valutazioni che vengono fornite ai prodotti da esso venduto e un venditore con una bassa valutazione viene giudicato come poco affidabile. Di solito le richieste dei venditori di effettuare recensioni false vengono fatte tramite annunci presenti su diversi canali telegram. Abbiamo deciso di concentrarci su Amazon perchè è la piattaforma per cui sono presenti più annunci in assoluto. Amazon è un'azienda di commercio elettronico statunitense, con sede a Seattle nello stato di Washington. È la più grande Internet company al mondo. Time Magazine ha proclamato Jeff Bezos, fondatore dell'azienda, Uomo dell'anno del 1999, a riconoscimento del successo di Amazon nel rendere popolare il commercio elettronico [11]. La forte presenza di annunci in cui si richiedono recensioni false su Amazon è probabilmente dovuta alla sua enorme importanza e al fatto che dà molto rilievo alle recensioni, perché comunque essendo una piattaforma con un elevato numero di utenti è difficile stare dietro a chiunque quindi, per garantire la qualità della merce, invece che effettuare interventi successivi alla vendita, prova a limitare i reclami andando a imporre degli standard di qualità ai prodotti definiti dagli stessi acquirenti tramite le valutazioni. Gli stessi utenti della piattaforma, che siano essi acquirenti o venditori, hanno una loro valutazione che ne definisce la credibilità di ciò che dicono, nel caso dell'acquirente, o vendono, nel caso del venditore. La credibilità su questo tipo di piattaforma è tutto come si può capire perché, giustamente, un venditore con una bassa credibilità ottenuta per via di valutazioni su prodotti precedentemente venduti di scarsa qualità, riuscirà a vendere meno perchè i prossimi acquirenti si fideranno meno. Ed è qui che nasce la richiesta di effettuare recensioni false, che possano rendere quel venditore il più credibile possibile in modo da poter "truffare" i suoi successivi clienti. Gli utenti che invece sono sulla piattaforma solo per acquistare perché dovrebbero effettuare recensioni dopo aver acquistato il prodotto? E cosa li spinge a fare recensioni false? Effettivamente l'utente non avrebbe nessuna motivazione nel fare recensioni se non voler dare il proprio parere su un qualcosa che si è acquistato solo per aiutare i successivi compratori ad avere un giudizio, ma questo non basta. Amazon ha quindi ideato un metodo per poter incentivare le persone a valutare i prodotti, che è la classifica dei top recensori. Un utente Amazon ha un proprio posto in questa classifica che aumenta in base al numero di recensioni effettuate e alla qualità di queste, definita in

base agli utili ricevuti, che sono molto simili ai “mi piace” che troviamo in diversi social network. Ma la motivazione non è data solo dall’ essere primo tra tanti, ma dal fatto che più si è alti in classifica e più si è credibili come recensori, questa credibilità risalta molto agli occhi degli altri acquirenti. Per queste motivazioni una recensione di un utente alto in classifica vale molto per chi vende, è “buona pubblicità” e diversi venditori sulla piattaforma si affidano a chiunque per effettuare una recensione del proprio prodotto, regalando il prodotto stesso. Questo genera un ulteriore fenomeno che è quello di portare le persone a far crescere il proprio account nella classifica recensori per accedere a prodotti sempre più costosi in cambio di una semplice recensione e quindi si trovano su telegram anche annunci di persone che richiedono follower o utili. Insomma, su Amazon come anche su altre piattaforme, dove il concetto di affidabilità è fondamentale per vendere viene messo a dura prova da persone che cercano di sembrare ciò che non sono, solo per un guadagno personale. Siccome tutta questa “truffa” ruota attorno alle recensioni, visto che queste rappresentano in fin dei conti il parametro su cui viene valutata una persona o un prodotto, con questo progetto puntiamo a valutare la veridicità di una recensione, in modo da poterla classificare come falsa o vera e di conseguenza capire se poter prendere sul serio o meno.

Il linguaggio che useremo per svolgere questo tipo di progetto sarà Python. Per prima cosa prendiamo in considerazione un canale telegram contenente annunci di prodotti su cui i venditori chiedono una recensione fake in cambio del prodotto. Attraverso la creazione e l’uso di un bot telegram vengono raccolti tutti i link dei prodotti nel momento stesso in cui vengono inseriti sul canale. Da ogni prodotto raccoglieremo informazioni sia riguardanti tutte le recensioni, come nome del cliente, numero di stelle date ecc... raccoglieremo anche le informazioni riguardanti il venditore, come il nome o gli indirizzi utilizzati. Ottenute poi tutte le recensioni svolgeremo una fase di data cleaning, eliminando le immagini ed effettuando una pulitura riguardante la punteggiatura, cancellazione degli spazi e piccole correzioni grammaticali. Corretti i dati, per analizzarli e insegnare alla macchina a riconoscere una recensione falsa da ciò che ha a disposizione, abbiamo deciso di utilizzare Bert [12] per il Natural Language Processing poiché esiste un maggior numero di documenti online e codice disponibile a differenza di altri, come MUSE.

Il seguente documento si divide in related work, nel quale spieghiamo da chi e da dove andiamo a recuperare le informazioni, citando altri ricercatori o sviluppatori. Il capitolo successivo spiegheremo come è avvenuta la creazione dei dataset e delle scelte effettuate per l'etichettatura delle recensioni. Negli ultimi due capitoli avremo una parte riguardante la metodologia proposta e le valutazioni dei modelli sviluppati, e una conclusione.

2) Related Work

Come descritto in precedenza, in questo articolo affrontiamo il problema dell'identificazione di recensioni false su Amazon. L'individuazione di recensioni false riguarda l'analisi di parametri legati alla recensione stessa, come la data di pubblicazione, la lunghezza e l'idea è quella di individuare anche delle caratteristiche comportamentali e semantiche relative al modo in cui i clienti scrivono la recensione, utilizzando tecniche di apprendimento automatico, ovvero supervisionato. Tra i lavori che considerano caratteristiche testuali in [2] vengono illustrati alcuni approcci che utilizzano tecniche di apprendimento automatico e feature esclusivamente linguistiche. In [3] propongono un approccio basato su tecniche di apprendimento automatico supervisionato in cui, le feature coinvolte comprendono sia attributi legati al contenuto testuale della recensione sia al profilo dell'utente. Le valutazioni vengono condotte su dataset etichettati, ovvero insiemi di dati la cui credibilità sia nota anche se non esistono dataset di review in cui la classificazione in veritiere o meno possa essere garantita come affidabile al 100%. In [3] le recensioni vengono classificate manualmente ma ciò non permette di creare uno standard di dimensioni significative per le analisi. Il nostro studio cerca di combinare diverse tecniche, andando ad analizzare innanzitutto i dati di utenti e prodotti come nel caso di [3] per fare una iniziale valutazione delle review, definendo un dataset iniziale nel quale, ovviamente, anche nel nostro caso, non possiamo averne una credibilità del 100%, ma considerando che partiamo da prodotti trovati all'interno di canali telegram nei quali si richiedono recensioni false, in teoria sappiamo che esistono e cerchiamo attraverso un sistema di classificazione basato su parametri che appartengono di norma a recensioni fake di aumentare l'affidabilità dell'etichettatura iniziale. Questo approccio di effettuare una classificazione automatica iniziale per definire un dataset è di per sé una nostra peculiarità, visto che in [3] questa cosa viene fatta solo per etichettare e non per definire dataset di studio. Un altro argomento che ci differenzia da altri articoli che vanno ad allenare modelli supervisionati è che la classificazione è stata effettuata con una tecnica particolare di classificazione del testo (NLP) lavorando attraverso un modello pre-addestrato (Bert) che ci permette di individuare delle similitudini nel testo, tecnica che viene utilizzata soprattutto in un contesto social per individuare i sentimenti degli utenti tramite la lettura dei post [4], nella sua versione italiana, che ci risulta essere molto innovativa visto l'attuale poco utilizzo di Bert, soprattutto della sua versione italiana visto la sua natura ancora sperimentale.

3) Dataset

Per raccogliere i dati da inserire nei dataset è stato sviluppato un crawler di cui presenteremo la struttura in maniera dettagliata nel capitolo successivo, in questo capitolo ci concentreremo sul descrivere la struttura dei dataset.
I dati raccolti riguardano il prodotto, le recensioni del prodotto e il venditore. Con queste informazioni sono stati realizzati due dataset: “review” e “AmazonDatasetSeller”. Il primo dataset “review” riguarda le recensioni e qui di seguito possiamo osservare la struttura di tale documento.

Utente	Posizione_utente	Prodotto	Titolo	Contenuto	Rate	Media	Data	Voti
twixi	Verificato	/royal-canin-Hypoalle	Mangime umido deli	Nostra cagnolina be	5,0 su 5 stelle	3,9 su 5	Recensito in Italia il	0
VITTORIO Z.	Oltre i 250000	/royal-canin-Hypoalle	Prodotto valido	Arrivato nei tempi p	4,0 su 5 stelle	3,9 su 5	Recensito in Italia il	0
Daniela1987	Oltre i 250000	/royal-canin-Hypoalle	ottimo prodotto	ottimo cibo umido	5,0 su 5 stelle	3,9 su 5	Recensito in Italia il	Una persona l'ha trov
Massimo	881	/royal-canin-Hypoalle	Royal canin hypoalle	Ottimo cibo per cat	5,0 su 5 stelle	3,9 su 5	Recensito in Italia il	0

La colonna “Utente” contiene il nickname degli utenti che hanno fatto la recensione mentre la posizione utente indica la sua posizione nella classifica dei top recensori o se l’utente è verificato, perché se hai un account verificato non è posizionato nella classifica. Nel dataset si può osservare che fino alla posizione duecentocinquantamila è visibile il numero preciso della posizione che un utente occupa in graduatoria, oltre viene visualizzato semplicemente una stringa con su scritto “oltre i 250000” perchè la posizione dall’account utente con il crawler non è recuperabile. Quindi è stata estrapolata la classifica dal sito Amazon e ogni volta viene ricercata la posizione occupata dal relativo utente e, per questioni di tempo, la classifica generale è stata estrapolata fino a duecentocinquantamila. La colonna “Prodotto” contiene il link del prodotto preso in considerazione e le successive riguardano le recensioni, ovvero, il titolo della recensione, il contenuto, il numero di voti di utilità ottenuto da una specifica recensione, il punteggio che ha espresso il recensore e la data di pubblicazione della recensione. Infine, è stato inserito anche come parametro anche la media delle stelle ottenute dal prodotto.
Una volta realizzato il dataset abbiamo svolto una fase di data cleaning su di esso, più precisamente sul contenuto della recensione, andando ad applicare tali correzioni attraverso l’utilizzo delle regular expression:

- la recensione è stata pulita dalla punteggiatura
- è stata corretta la grammatica da parole come “nn”, “ke”, attraverso l’utilizzo di un dizionario che, ogni volta che trova una parola come quelle descritte in precedenza, le sostituisce con le parole corrette come “non” e “che”
- tolti gli spazi maggiori di uno
- tolte tutte le emoji ed emoticons
- infine, reso tutto il testo in minuscolo.

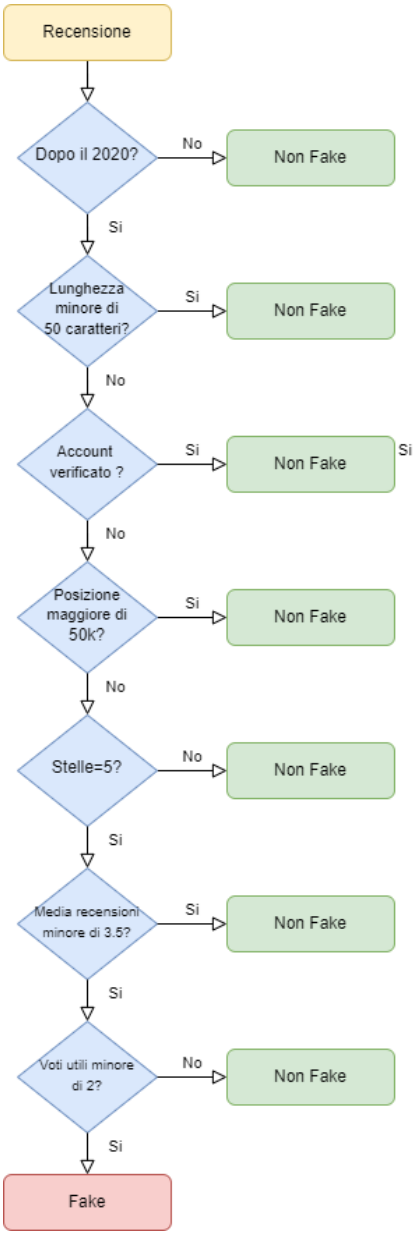
Con la pulizia sui dati abbiamo creato un nuovo documento chiamato “reviewCleaned” e qui di seguito possiamo osservare la struttura di tale documento.

Utente	Posizione_utente	Prodotto	Titolo	Contenuto	Rate	Media	Data	Voti	Classe
twixl	Verificato	/royal-canin-Hypoall	Mangime umido del	nostra cagnolina bea	5	3.9	Recensito in Italia il	0	0
VITTORIO Z.	Oltre i 250000	/royal-canin-Hypoall	Prodotto valido	arrivato nei tempi pr	4	3.9	Recensito in Italia il	0	0
Daniela1987	Oltre i 250000	/royal-canin-Hypoall	ottimo prodotto	ottimo cibo umido p	5	3.9	Recensito in Italia il	1	0
Massimo	881	/royal-canin-Hypoall	Royal canin hypoall	ottimo cibo per cani	5	3.9	Recensito in Italia il	0	0
Giulio11	Oltre i 250000	/royal-canin-Hypoall	Abbastanza buono	un buon prodotto noi	4	3.9	Recensito in Italia il	0	0

Da qui siamo passati a classificare le recensioni secondo se esse fossero reali o false, valutando i dati a nostra disposizione. La classificazione della review ha tenuto conto dei seguenti parametri, nel caso tutte fossero rispettate la review viene classificata come falsa:

- Se ci sono più di 50 parole nel testo, visto che le recensioni false spesso contano un minimo di parole richieste spesso dallo stesso venditore.
- Se l’account non è verificato e la sua posizione non è maggiore di cinquantamila, un account verificato potrebbe rischiare di perdere i propri privilegi per fare delle recensioni fake mentre, chi è oltre una determinata posizione è considerabile un utente comune che non utilizza molto fare recensioni e quindi le poche che fa sono considerabili reali.
- Se la recensione ha cinque stelle, ovvero il parametro richiesto dal venditore negli annunci per aumentare la propria affidabilità.
- Se il numero medio di stelle di un prodotto è minore di 3.5 e le stelle sono cinque, perché in questo caso si tratta di un prodotto che la maggior parte degli utenti hanno valutato come scadente mentre, un utente che dà 5 stelle quasi sicuramente lo avrà fatto come “favore” al venditore.
- Se il numero di voti utili è minore di due, perchè spesso gli utenti che scrivono questo tipo di recensioni richiedono su certi canali anche di mettere il “like” alla review in modo da aumentare la credibilità di questa e permette al recensore anche di salire in classifica.
- Se la review è stata fatta prima del 2020, poiché il fenomeno delle recensioni false è nato nel 2019 e diffuso nel 2020, consideriamo non fake tutte le recensioni prima del 2020.

Per far capire meglio il nostro metodo di classificazione potete notare il flowchart sottostante nel quale vengono spiegati tutti i passaggi per classificare una recensione fake da una non fake.

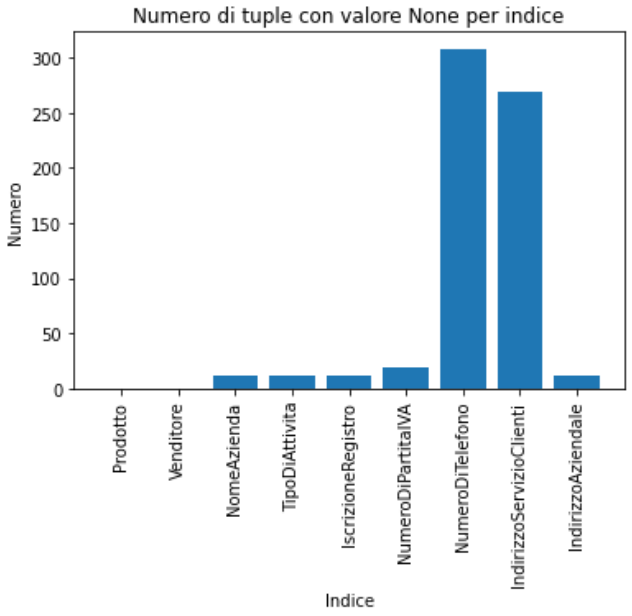


Quindi a seconda dei valori di tali parametri la classe a cui è stata associata la recensione è 1 nel caso in cui corrisponde ad una recensione falsa, oppure 0.
Il dataset “AmazonDatasetSeller” invece contiene solo le informazioni riguardanti i venditori. Le colonne del dataset contengono il nome del prodotto, il venditore, il nome dell’azienda, numero di iscrizione al registro delle imprese, il numero di partita IVA, il numero di telefono, Indirizzo Servizio clienti, Indirizzo aziendale. Possiamo vedere dalla foto in basso la struttura del dataset.

Prodotto	Venditore	NomeAzienda	Tipo di attivita	Numero di iscrizione	Numero di partita IV	Numero di telefono	Indirizzo Servizio cli	Indirizzo aziendale
ROYAL CANIN	NutriPet	Nutripet di Varani Lu	Impresa privata	MN225887	IT02119530208	+390386841003	Via Abetone Brenne	Via Abetone Brenne
Cibocrudo Sem	CiboCrudo srl	CiboCrudo srl	Impresa privata	RM - 1413689	IT12965131001	774790100	Via Conte Roncone,	Via Conte Roncone,
Inodorina Salviette A	Acquatic Life SRL	Acquatic Life Srl	Impresa privata	MB-2613149	IT11594970961	+393489398751	None	Via Pizzo Scalino 2
PROBIOTICS 8	siQuri	Deva Srl	Impresa privata	None	IT05061680285	498380069	Via Visco 7LimenaP	Via Visco, 7Limenal

Questo dataset ha molti valori posti a “None” poiché i venditori non sono tenuti su Amazon a fornire pubblicamente tutti i dati e molti di questi dati risultano essere poco veritieri (come, per esempio, alcuni nomi di venditori), quindi molti venditori di questo dataset risultano essere account visibilmente falsi, creati unicamente per essere account dall’alta affidabilità ottenuta tramite il metodo delle recensioni false senza però rischiare di utilizzare le proprie informazioni reali.

Uno studio statistico che può essere fatto sui dati dei venditori riguarda la quantità di informazioni mancanti rispetto a quelle che sono convenzionalmente inserite. Il seguente diagramma ci mostra la totalità di valori assenti per ogni indice nel dataset e i parametri che mancano maggiormente riguardano proprio il contatto con il cliente, ovvero il “Numero di Telefono” e “Indirizzo Servizio Cliente” e questo ci può far pensare ad un venditore che cura poco il rapporto con la clientela di per sè, probabilmente, poco affidabile. A questo aggiungiamo che molte informazioni nel dataset, in particolare riguardo l’indirizzo aziendale, sono informazioni che non hanno un vero e proprio significato ma sono delle stringhe senza un particolare significato.



4) Metodologia Proposta per la creazione del modello predittivo

Per la raccolta dei dati è stato sviluppato un crawler, ovvero uno script che analizza automaticamente i contenuti presenti all’interno di sorgenti dati. Nel caso specifico del web, i web crawler sono bot utilizzati dai motori di ricerca per analizzare i contenuti all’interno di un sito e delle sue pagine, crearne un indice e permetterne la visualizzazione tra i risultati di ricerca. Per definire il crawler in questo specifico caso è stato utilizzato BeautifulSoup, una libreria di Python per analizzare le pagine web e recuperarne il contenuto. Gli usi comuni di BeautifulSoup includono la ricerca per

classe CSS, ricerca per indirizzo di collegamento ipertestuale, ricerca per ID elemento, ricerca per nome attributo e valore dell'attributo [10]. Attraverso l'utilizzo di un bot telegram sono stati raccolti i link dei prodotti per cui i venditori richiedono recensioni false attraverso annunci emessi su un canale telegram, inseriti in un file denominato "links". Il crawler, innanzitutto, legge dal documento "links" i link dei prodotti e li modifica andando a recuperare solo la parte successiva a "/dp", che contiene sostanzialmente il codice identificativo del prodotto su Amazon e aggiunge a questo il prefisso "https://www.amazon.it/product-reviews" che ci permette di avere il link della pagina delle revisioni di quel determinato prodotto. Un esempio è il seguente: dal link "https://www.amazon.it/dp/B009GY4NN6?tag=nexttech0922-21" togliamo la parte fino a /dp, ovvero "/B009GY4NN6?tag=nexttech0922-21", prendiamo quindi la stringa citata in precedenza riguardante le recensioni del prodotto e la aggiungiamo alla stringa di riferimento del prodotto così da ottenere "https://www.amazon.it/product-reviews/B009GY4NN6?tag=nexttech0922-21".
A questo poi aggiungiamo il suffisso "&pageNumber=x" dove x indica un numero, poiché la pagina dei commenti per ogni prodotto non è unica, essendo che in una pagina di recensioni possono esserci massimo dieci recensioni. Dai link costruiti con prefisso e suffisso accediamo quindi ad una pagina contenente unicamente recensioni di un determinato prodotto e tramite la funzione soup.select("path"), dove "path" rappresenta il percorso css identificativo di un determinato elemento all'interno della pagina, recuperiamo le informazioni. La prima informazione che recuperiamo è il link della singola recensione, salviamo tutti i link in una struttura e successivamente tramite questi link accediamo alla pagina dedicata della specifica recensione, da qui, sempre tramite la stessa funzione, vengono recuperate tutte le informazioni per ogni recensione dove è presente un link nella struttura sopracitata. Quindi, a questo punto, ci ritroviamo una serie di strutture contenenti le informazioni delle recensioni presenti all'interno della pagina definita precedentemente, queste informazioni vengono salvate all'interno del file "review" insieme al link del prodotto a cui sono associate le recensioni. Per quanto riguarda le informazioni del venditore è stato costruito un crawler simile a quello descritto in precedenza però con diversi controlli riguardanti il fatto che un venditore non è tenuto a mostrare tutti i suoi dati.

Dopo la generazione dei dataset come descritto in precedenza siamo passati all'analisi dei dati per definire un modello predittivo per riconoscere le revisioni fake, generato attraverso l'algoritmo di apprendimento NLP con l'ausilio di Bert. Il Natural Language Processing (NLP) o elaborazione del linguaggio naturale è un ramo dell'intelligenza artificiale che si occupa di fornire ai computer la capacità di comprendere il testo, come gli esseri umani. L'elaborazione del linguaggio naturale comporta un processo particolarmente delicato a causa delle complesse caratteristiche del linguaggio stesso [5]. Per quanto riguarda la struttura di Bert, si tratta della prima architettura nel Natural Language Processing ad essere pre-addestrata, utilizzando un apprendimento non supervisionato su puro testo non strutturato (circa 2,5 miliardi di parole). BERT utilizza il Transformer, un meccanismo che apprende le relazioni contestuali tra le parole in un testo. Nella sua forma

standard, il Transformer include due meccanismi separati: un codificatore che legge l'input di testo e un decodificatore che produce una previsione. Poiché l'obiettivo di BERT è generare un modello di linguaggio, è necessario solo il meccanismo dell'encoder. A differenza dei modelli direzionali, che leggono l'input di testo in sequenza (da sinistra a destra o viceversa), il codificatore Transformer legge l'intera sequenza di parole contemporaneamente. Questa caratteristica consente al modello di apprendere il contesto di una parola in base a tutto ciò che lo circonda. L'input è una sequenza di token, che vengono prima incorporati nei vettori e quindi elaborati nella rete neurale. L'output è una sequenza di vettori di dimensione H, in cui ogni vettore corrisponde a un token di input con lo stesso indice. Prima di inserire sequenze di parole in BERT, il 15% delle parole in ciascuna sequenza viene sostituito con un token, detto MASK. Il modello tenta quindi di prevedere il valore originale delle parole mascherate, in base al contesto fornito dalle altre parole non mascherate nella sequenza. Vengono prese in considerazione solo le previsioni dei valori mascherati e ignorate le previsioni delle parole non mascherate. Nel processo di addestramento, il modello riceve coppie di frasi come input e impara a prevedere se la seconda frase nella coppia è la frase successiva nel documento originale. Durante l'addestramento, il 50% degli input sono una coppia in cui la seconda frase è la frase successiva nel documento originale, mentre nell'altro 50% viene scelta come seconda frase una frase casuale del corpus. Il presupposto è che la frase casuale sarà scollegata dalla prima frase. Per aiutare il modello a distinguere tra le due frasi in addestramento, l'input viene elaborato nel modo seguente prima di entrare nel modello:

1. Un token [CLS] viene inserito all'inizio della prima frase e un token [SEP] viene inserito alla fine di ogni frase.
2. A ciascun token viene aggiunta una frase incorporata che indica la Frase A o la Frase B.
3. Un'inclusione posizionale viene aggiunta a ciascun token per indicarne la posizione nella sequenza.

Un esempio di tokenization lo possiamo osservare sulle frasi "questo è un tutorial modello bert" e "metteremo a punto un modello bert" che vengono convertite nel seguente modo.

['[CLS]', 'questo', 'è', 'un', 'tu', '##toria', '##l', 'modello', 'ber', '##t', '[SEP]', 'mettere', '##mo', 'a', 'punto', 'un', 'modello', 'ber', '##t', '[SEP]']

Per prevedere se la seconda frase è effettivamente collegata alla prima, si eseguono i seguenti passaggi:

1. L'intera sequenza di input passa attraverso il modello Transformer.

2. L'output del token [CLS] viene trasformato in un vettore sagomato 2×1, utilizzando un semplice livello di classificazione (matrici apprese di pesi e bias).
3. Calcolo della probabilità con softmax [9].

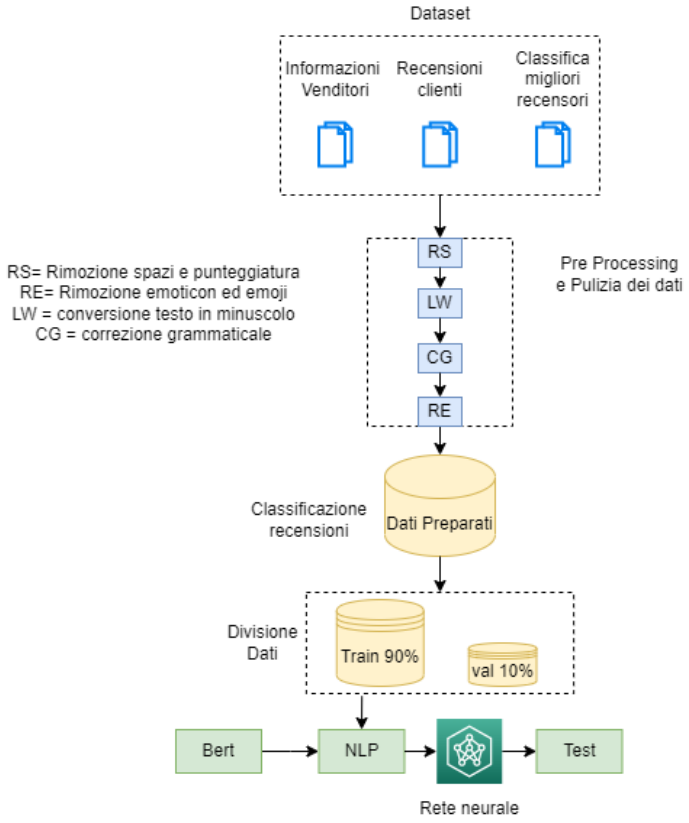
Per implementare questo tipo di architettura è necessario effettuare una fase di fine-tuning per perfezionare task specifici sul linguaggio naturale [6]. L'elaborazione di un testo inizia con la sua scomposizione in token corrispondenti a spazi, parole, punteggiatura, frasi. Il task non è particolarmente complesso ma presenta comunque alcune problematiche: per esempio, se si considera il punto come fine di una frase, si rischia di sbagliare frequentemente in quanto il punto potrebbe riferirsi a una abbreviazione, a una data o a un link [5].

Per quanto riguarda invece la parte implementativa è stato utilizzato un modello pre-addestrato di Bert, andando a cercare in particolare la sua versione in lingua italiana[7][13]. Abbiamo importato il modello pre-addestrato "dbmdz/bert-base-italian-xxl-cased" che in [7][13] risultava essere il migliore.
Tale processo è detto transfer learning e sono due le possibili soluzioni:

1. Estrazione delle caratteristiche: usa le rappresentazioni apprese da una rete precedente per estrarre caratteristiche significative da nuovi campioni. È sufficiente aggiungere un nuovo classificatore, che verrà addestrato da zero, al di sopra del modello pre-addestrato in modo da poter riutilizzare le mappe delle caratteristiche apprese in precedenza per il set di dati.
2. Fine-tuning: sblocca alcuni dei livelli superiori di una base del modello congelata e addestra insieme sia i livelli classificatori appena aggiunti che gli ultimi livelli del modello base. Questo ci consente di "regolare" le rappresentazioni delle caratteristiche di ordine superiore nel modello base per renderle più rilevanti per l'attività specifica.

Noi abbiamo utilizzato il primo metodo perché la rete di base contiene già caratteristiche che sono generalmente utili per classificare. Tuttavia, la parte finale di classificazione del modello pre-addestrato è specifica dell'attività di classificazione originale e successivamente specifica dell'insieme di classi su cui è stato addestrato il modello. Quindi freeziamo tutti i livelli e aggiungiamo due layer convoluzionali da addestrare.Recuperiamo dai dataset il contenuto delle recensioni che andiamo a dividere tra train set e validation set, successivamente tokenizzate e codificate per poi convertire le sequenze di token in tensori. Per migliorare l'allenamento della rete neurale è stata implementata la loss function andando ad assegnare dei pesi alle classi, assegnando alla classe 0 un peso di 0.53 e alla classe 1 un peso di 6.75, per poter bilanciare la mancanza di campioni di classe 1. Infine, è stato automatizzato il processo di addestramento che ci permetterà di addestrare le reti neurali cambiando in automatico gli iperparametri che vedremo nel prossimo capitolo. Nell'immagine seguente possiamo vedere come è stata strutturata l'intera architettura del nostro

progetto, guardando il seguente diagramma dall'alto verso il basso. Nella prima fase viene effettuata l'estrazione delle informazioni riguardanti gli utenti e i venditori associati ai prodotti, la seconda fase riguarda la pulizia e la classificazione dei dati tramite gli standard da noi definiti e le ultime due fasi riguardano il processo di divisione dei dati in train, test e validation e successivamente l'utilizzo di nlp con bert per generare una rete neurale che possa classificare le informazioni secondo le indicazioni e infine il test della rete generata.

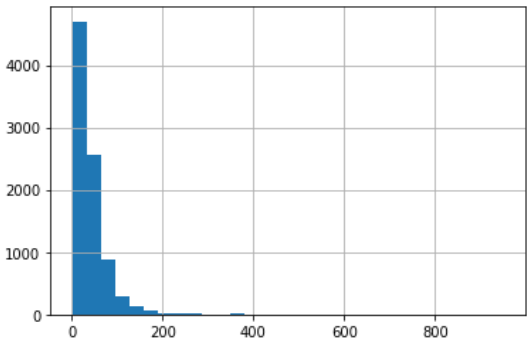


5) Valutazione Sperimentale

In questa fase trattiamo i risultati ottenuti dai test effettuati, di come abbiamo lavorato e tutti gli iperparametri modificati per ogni tipo di allenamento. Ogni modello generato durante questa fase è stato testato con un unico test oracolo che presenta informazioni veritiere, non sono molti valori ma comunque siamo certi che sono informazioni certe, visto che sono state fatte a mano e non seguendo un metodo di etichettatura automatico sperimentale come per i dati che utilizziamo per gli allenamenti e questo ci aiuterà a capire di più le prestazione dei modelli generati.

Il grafico sottostante indica la lunghezza delle recensioni all'interno del data set, la lunghezza è una caratteristica molto importante al fine di definire un modello, poiché quando generiamo le sequenze di token è importante capire effettivamente quanto

queste sequenze debbano essere lunghe per evitare di perdere informazioni troncando le frasi con sequenze piccole.



La scelta migliore sarebbe andare a considerare una lunghezza massima delle sequenze lunghe tanto quanto la stringa più lunga, troncando poi le sequenze di zeri create per riempire le stringhe più corte ma, come possiamo osservare la stringa più lunga arriva a circa 400 caratteri e questo avrebbe portato a dover fare diversi tentativi di addestramento che avrebbero richiesto molto tempo e risorse, visto che con l'aumentare della lunghezza delle sequenze di token aumenta il tempo necessario per l'analisi. Di conseguenza abbiamo preferito effettuare degli allenamenti impostando la lunghezza delle sequenze come il numero medio della lunghezza di tutte le stringhe ovvero 40 e con la lunghezza tra le recensioni preponderante, ovvero 25.

I dati sono stati divisi inizialmente in 85% dei dati per il train e il 15% per i dati di validation, aggiungendo un parametro che ci ha permesso inoltre di bilanciare i dati.
Con i dati del train sono stati effettuati gli allenamenti andando innanzitutto a modificare due iperparametri come il numero di epoche, che definisce il numero di volte in cui l'algoritmo di apprendimento funzionerà attraverso l'intero set di dati di addestramento, e la grandezza della batch, ovvero la dimensione che definisce il numero di campioni che verranno propagati attraverso la rete. La tabella sottostante rappresenta tutti i test effettuati con relativi parametri e indichiamo anche i risultati di accuratezza, precision e recall per ogni classe. La tabella di seguito riporta i risultati dei test dei modelli prodotti con tokenizzazione a 25 e 40 senza l'applicazione della loss function pesata sui dati.

Learning_rate	Epoche	Batch_size	Accuracy	Precision_0	Precision_1	Recall_0	Recall_1
0.001	5	8	0.56	0.56	0	1	0
0.001	5	16	0.56	0.56	0	1	0
0.001	5	32	0.56	0.56	0	1	0
0.001	5	64	0.56	0.56	0	1	0
0.001	5	128	0.56	0.56	0	1	0
0.001	5	256	0.56	0.56	0	1	0
0.001	10	8	0.56	0.56	0	1	0
0.001	10	16	0.56	0.56	0	1	0
0.001	10	32	0.56	0.56	0	1	0

0.001	10	64	0.56	0.56	0	1	0
0.001	10	128	0.56	0.56	0	1	0
0.001	10	256	0.56	0.56	0	1	0
0.001	15	8	0.56	0.56	0	1	0
0.001	15	16	0.56	0.56	0	1	0
0.001	15	32	0.56	0.56	0	1	0
0.001	15	64	0.56	0.56	0	1	0
0.001	15	128	0.56	0.56	0	1	0
0.001	15	256	0.56	0.56	0	1	0
0.001	20	16	0.56	0.56	0	1	0
0.001	20	32	0.56	0.56	0	1	0
0.001	20	64	0.56	0.56	0	1	0
0.001	25	16	0.56	0.56	0	1	0
0.001	25	32	0.56	0.56	0	1	0
0.001	25	64	0.56	0.56	0	1	0
0.01	20	16	0.56	0.56	0	1	0
0.0001	20	16	0.56	0.56	0	1	0
0.01	25	16	0.56	0.56	0	1	0
0.0001	25	16	0.56	0.56	0	1	0
0.002	20	16	0.56	0.56	0	1	0
0.003	20	16	0.56	0.56	0	1	0
0.002	25	16	0.56	0.56	0	1	0
0.003	25	16	0.56	0.56	0	1	0

La seguente tabella è unica per entrambi gli esperimenti, ovvero con sequenze lunghe 25 e 40 poiché i risultati sono gli stessi, e possiamo notare che malgrado la diversità dei parametri cambiati per gli allenamenti i modelli restituiscono come risultato dei test sempre una Recall_0 pari a 1, questo vuol dire che la percentuale di elementi effettivamente presenti nell'input che sono stati correttamente identificati dal sistema con classe zero è del 100%, quindi per la classe zero questi tipi di allenamento sono ottimi ma, d'altra parte, la Recall_1 è sempre 0, indicando che il sistema non riesce mai a predire che una determinata recensione è fake. Probabilmente questi valori sono dovuti al fatto che i falsi negativi e i veri negativi sono uguali a 0 ciò si traduce nel fatto che il sistema non restituisce mai il valore 1 come previsione ma assegna sempre 0 e probabilmente è dovuto al fatto che il sistema non riesce a generalizzare bene la classe 1 visto la scarsità dei dati a sua disposizione. Quindi il passo successivo è stato quello di ripetere gli allenamenti con la stessa sequenza di iperparametri della tabella precedente ma andando ad utilizzare la loss function per bilanciare meglio i dati degli allenamenti visto che la scarsità di recensioni di classe 1 ci ha portato ai pessimi risultati precedentemente mostrati. Nella successiva tabella mostriamo ora i migliori risultati in base all'accuracy, riguardanti gli allenamenti con loss function e tokenizzazione a 25.

Learning_rate	Epoche	Batch_size	Accuracy	Precision_0	Precision_1	Recall_0	Recall_1
0.002	25	16	0.61	0.7	0.53	0.58	0.65
0.001	15	16	0.6	0.7	0.51	0.53	0.69
0.001	25	16	0.6	0.62	0.53	0.78	0.35
0.002	20	16	0.6	0.68	0.52	0.58	0.62
0.003	25	16	0.6	0.64	0.52	0.69	0.46
0.001	5	8	0.58	0.58	0	1	0
0.001	10	8	0.58	0.58	0	1	0
0.001	15	8	0.58	0.58	0	1	0
0.0001	20	16	0.58	0.58	0	1	0
0.001	5	32	0.55	0.6	0.45	0.67	0.38

Dalla tabella notiamo che i risultati si discostano da quelli precedenti riguardanti il fatto di avere sempre lo stesso valore e soprattutto sulla recall ora la maggior parte non ci restituisce valori unitari, di conseguenza possiamo dire che con la loss function pesata riusciamo a generalizzare meglio la classe delle recensioni fake. Il fatto di aver implementato la loss function però non ci ha permesso di ottenere risultati ottimi. Quelli evidenziati in arancione sono i casi peggiori per le stesse motivazioni per cui i modelli senza loss function non erano adatti infatti notiamo i valori della recall unitari. I migliori risultati sono evidenziati in verde e quello che notiamo è che, nonostante questi abbiano una percentuale di accuracy oltre il 50%, su un test pressoché piccolo, quindi non così negativa, il fatto che la precision sia in entrambi i casi, ovvero sia per la classe 0 che per la classe 1, molto vicina al 60% non ci permette di validare questi modelli come modelli ottimali. Di seguito mostriamo anche i risultati ottenuti dai test dei modelli effettuati con la stessa tecnica tranne che per la lunghezza delle sequenze di token impostata a 40.

Learning_rate	Epoche	Batch_size	Accuracy	Precision_0	Precision_1	Recall_0	Recall_1
0.001	5	8	0.6	0.59	1	1	0.04
0.001	10	8	0.6	0.59	0.67	0.97	0.08
0.003	20	16	0.6	0.82	0.51	0.39	0.88
0.001	10	32	0.58	0.86	0.5	0.33	0.92
0.001	15	8	0.58	0.58	0	1	0
0.001	15	16	0.56	0.67	0.49	0.5	0.65
0.001	25	16	0.55	0.65	0.47	0.47	0.65
0.0001	25	16	0.55	0.67	0.47	0.44	0.69
0.001	5	32	0.53	0.59	0.43	0.67	0.35
0.001	15	32	0.53	0.89	0.47	0.22	0.96

In questo caso notiamo che i due casi con accuracy maggiore non possono essere considerati perchè guardando la Recall_1 notiamo valori molto bassi, questo ci indica che il sistema spesso restituisce dei falsi negativi, ovvero nel momento in cui dovrebbe assegnare 1 ci restituisce 0. Questa cosa ci dice che il sistema nonostante

una precisione unitaria della classe 1, ovvero una precisione assoluta nell'indicare che una recensione fake lo sia effettivamente, si espone molto raramente su questo tipo di classe restituendo spesso la classe 0. I casi migliori sono nuovamente evidenziati in verde e in questo caso, rispetto a prima, notiamo una precisione sulla classe 0 più elevata quindi, questi modelli sono considerabili migliori dalla loro precisione che migliora sulla classe 0 e che rimane simile sulla precisione della classe 1. Ma quello che a noi interessa maggiormente nel nostro dominio non è tanto il valore della precision, ovvero l'affidabilità di quello che ci viene detto, ma ci focalizziamo maggiormente sul valore della recall, perchè a noi interessa che il sistema riesca il più possibile ad individuare l'etichettatura reale del test. Anche osservando questo parametro però possiamo dire che questi nuovi modelli sono migliori perchè una recall più alta rispetto a prima ci indica che ci sono meno falsi negativi e quindi più valori che il sistema ha effettivamente individuato come quelli reali.

Abbiamo proseguito con la sperimentazione provando a dare percentuali diverse per dividere i dati per l'allenamento, cambiando la percentuale di train da 85 a 90 e di conseguenza quella di validation da 15 a 10, questo perchè volevamo testare dei modelli prodotti da un numero superiore di dati di train, idea nata dal fatto che il bilanciamento della loss function ci ha portato risultati migliori, di conseguenza il pensiero è stato che facendo lavorare la fase di train con un maggior numero di dati rispetto a prima e mantenendo la loss function magari i risultati potevano essere migliori. Di seguito è possibile osservare la tabella che mostra i migliori risultati dei modelli allenati sempre con gli stessi parametri definiti in precedenza con lunghezza delle sequenze a 25 e, come detto, con le percentuali di suddivisione dati cambiate.

Learning_rate	Epoche	Batch_size	Accuracy	Precision_0	Precision_1	Recall_0	Recall_1
0.001	20	32	0.61	0.8	0.52	0.44	0.85
0.003	25	16	0.61	0.75	0.53	0.5	0.77
0.001	5	8	0.58	0.58	0	1	0
0.001	10	8	0.58	0.58	0	1	0
0.001	15	8	0.58	0.58	0	1	0
0.0001	25	16	0.58	0.58	0	1	0
0.001	25	32	0.56	0.91	0.49	0.28	0.96
0.01	25	16	0.56	0.61	0.48	0.69	0.38
0.001	5	16	0.55	0.61	0.46	0.64	0.42
0.001	20	16	0.55	0.61	0.46	0.61	0.46

I migliori risultati sono evidenziati in verde e lo possiamo notare non solo dall'accuratezza ma anche in base alla precision e alla recall poiché, come abbiamo già spiegato, una precisione alta ci permette di capire quanto è affidabile ciò che dice il modello, comunque notiamo esser abbastanza alte, e la recall che ci indica l'accuratezza verso la singola classe tendono ad essere per entrambe le classi valori

accettabili. Nonostante ciò, ci sentiamo di dire che i modelli degli allenamenti precedenti con le sequenze lunghe 40 sono migliori, forse dovuto alla lunghezza delle stesse, quindi abbiamo effettuato l'allenamento con la nuova suddivisione dei dati anche con questo tipo di lunghezza. Mostriamo ora i migliori risultati nella tabella successiva

Learning_rate	Epoche	Batch_size	Accuracy	Precision_0	Precision_1	Recall_0	Recall_1
0.001	15	8	0.58	0.58	0	1	0
0.001	5	8	0.56	0.57	0	0.97	0
0.001	10	8	0.56	0.57	0	0.97	0
0.001	15	16	0.56	0.67	0.49	0.5	0.65
0.0001	25	16	0.55	0.83	0.48	0.28	0.92
0.001	10	16	0.53	0.63	0.46	0.47	0.62
0.001	10	32	0.53	0.68	0.47	0.36	0.77
0.001	10	128	0.53	1	0.47	0.19	1
0.001	20	32	0.53	0.82	0.47	0.25	0.92
0.001	25	16	0.53	0.64	0.46	0.44	0.65

Mentre ci aspettavamo un effettivo miglioramento, viste le premesse, questo tipo di sperimentazione non ci ha portato grandi risultati, di fatti dove l'accuratezza tende ad essere migliore notiamo lo stesso problema che avevamo negli esperimenti iniziali ovvero valori di precision e recall unitari, o quasi. Quello che ci risulta essere il modello migliore evidenziato in verde non è comunque ottimale quanto quelli ottenuti in precedenza. Infine, vogliamo chiudere questa fase sperimentale parlando di quelli che sono i risultati ottenuti non dai modelli ma da uno studio del test oracolo con il nostro sistema di etichettatura automatica che avevamo utilizzato per classificare i dati utilizzati durante la fase di allenamento.

Accuracy	Precision_0	Precision_1	Recall_0	Recall_1
0.64	0.62	1	1	0.15

I valori che troviamo mostrano una situazione ottima per quanto riguarda l'individuazione di quelli che effettivamente sono le recensioni non fake ma non si può dire altrettanto di quelle che invece sono in realtà fake. Su 26 recensioni fake di fatti ne individua solo 4 mentre delle 36 recensioni non fake le individua tutte, il problema della poca precisione sulle non fake non è dovuto tanto al fatto che i parametri stabiliti per la loro individuazione non siano veritieri, infatti ne restituisce 4 ed effettivamente quelle sono realmente fake, il problema è la frequenza con cui questo tipo di etichettatura viene effettuata, probabilmente ciò è dovuto al fatto che i parametri tendono ad essere troppo restrittivi. Questo ci dice due cose, innanzitutto che il nostro sistema tende ad essere affidabile per quanto riguarda la veridicità della classificazione delle recensioni non fake ma d'altra parte il fatto che non ne individua sempre non lo rende effettivamente utile come sistema di etichettatura e la seconda

cosa è che non essendo di per sé un ottimo classificatore i dati che abbiamo utilizzato per allenare i modelli non sono effettivamente precisi e quindi questo influisce molto su quello che poi sono i risultati dei modelli, perchè apprendono da dati inesatti.

6) Conclusioni e Sviluppi futuri

Il nostro esperimento ha riguardato il fenomeno delle recensioni false sui siti web, nel nostro caso abbiamo analizzato il sito di e-commerce Amazon.it. La finalità di questo esperimento è stata di sviluppare un sistema che riuscisse a classificare le recensioni di un prodotto come false o non false a partire dal link del prodotto. Il lavoro per sviluppare il sistema è stato strutturato in diverse fasi. Nella prima fase troviamo la raccolta dei dati riguardanti le recensioni e i dati dei venditori dei prodotti. Successivamente abbiamo effettuato un lavoro di data cleaning sulle recensioni, in vista del fatto che le recensioni dovevano essere analizzate dall'algoritmo di NLP per individuare dei pattern per riconoscere il tipo di recensione, in questa stessa fase abbiamo etichettato le recensioni, con 0 o con 1, per indicare al sistema se sono recensioni false o meno, secondo determinate condizioni scelte a priori. La fase finale ha riguardato l'implementazione dell'algoritmo di NLP per studiare le recensioni utilizzando un modello pre-addestrato (BERT). Il lavoro ci ha portato ad ottenere dei modelli con i migliori risultanti aventi un'accuratezza media sul test effettuato di 0.60, risultato soddisfacente vista la poca quantità di valori da testare ma con diverse idee per poterlo migliorare in futuro. Una prima osservazione va fatta in merito alla quantità di dati disponibili perchè, anche se abbiamo raccolto oltre undicimila tuple, crediamo che con una maggiore quantità di dati il sistema probabilmente potrebbe comprendere meglio alcune dinamiche che lo porterebbero a classificare con un'accuratezza maggiore le recensioni. Un'ulteriore osservazione va fatta sul tipo di etichettatura effettuata prima dell'allenamento, poiché le recensioni sono state filtrate secondo parametri che accomunano le recensioni false, ma non danno la certezza che una data recensione sia falsa e, se una recensione in principio viene classificata in maniera errata, potrebbe influenzare negativamente l'allenamento, la scelta migliore sarebbe lavorare su un dataset con informazioni reali al 100%. Infine, segnaliamo come ulteriore miglioramento per il futuro una lavorazione maggiore del pre-processing come si dice in [5], dove oltre al tokenizer, si parla anche di altre tipologie di lavorazione sul testo che noi non abbiamo effettuato e che avrebbero probabilmente portato il sistema ad una migliore comprensione delle recensioni e ad una loro migliore classificazione.

Riferimenti

1. Jindal, N., Liu, B.: Opinion spam and analysis. In: Proceedings of the 2008 International Conference on Web Search and Data Mining, ACM (2008) 219–230
2. Ren, Y., Ji, D., Zhang, H.: Positive unlabeled learning for deceptive reviews detection. In: EMNLP. (2014) 488–498
3. Li, F., Huang, M., Yang, Y., Zhu, X.: Learning to identify review spam. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Volume 22. (2011) 2488
4. Marco Polignano and Pierpaolo Basile and Marco de Gemmis and Giovanni Semeraro and Valerio Basile: AlBERTo: Italian BERT Language Understanding Model for NLP Challenging Tasks Based on Tweets. InProceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019), CEUR Workshop Proceedings 2481
5.”Natural Language Processing, cos'è, come funziona e applicazioni”
6.”Cos'è Bert, l'algoritmo che cambia il mondo del Natural Language Processing - AI4Business”
7. “GitHub - stefan-it/italian-bertelectra: 🇮🇹 Italian BERT and ELECTRA models (incl. evaluation)”
8.”deep-learning-keras-tf-tutorial/BERT_email_classification-handle-imbalance.ipynb at master · codebasics/deep-learning-keras-tf-tutorial · GitHub”
9.“BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science ”
10.“beautifulsoup Tutorial => Iniziare con beautifulsoup”
11.“Amazon.com - Wikipedia”
12.“[1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”
13.“Stefan Schweter: Italian BERT and ELECTRA models, 2020, Zenodo, 10.5281/zenodo.4263142, bhttps://doi.org/10.5281/zenodo.4263142”