SENTIMENT ANALYSIS E TEXT CLASSIFICATION DELLE RECENSIONI DEI FARMACI

DELLA MURA DARIO[1] & DOCI DAVID[2] & FILIP SARA[3]

CONTENTS

ABSTRACT

In questo studio analizziamo il dataset "Drug Reviews" attraverso diverse tecniche di Text Mining. L'analisi si svolge in tre parti: la prima si concentra sulla fase di pre-processing del testo; la seconda parte è dedicata alla rappresentazione testuale, mentre nell'ultima parte sono implementati diversi strumenti per operare una Text Classification ed una Sentiment Analysis.
L'obiettivo dello studio è classificare le recensioni in positive o negative (o neutre) e valutare i risultati ottenuti con i molteplici approcci utilizzati.

[1] *793751, Università degli Studi di Milano-Bicocca, L.M. Data Science, A.A 2020/2021*
[2] *799647, Università degli Studi di Milano-Bicocca, L.M. Data Science, A.A 2020/2021*
[3] *852864, Università degli Studi di Milano-Bicocca, L.M. Data Science, A.A 2020/2021*

1 INTRODUZIONE

L'obiettivo del seguente progetto è applicare la classificazione testuale alle recensioni provenienti dal dataset "Drug Reviews". In particolare si tratta di una classificazione binaria (ad eccezione di Bert), poiché ogni recensione viene assegnata ad una categoria tra "positiva" e "negativa".
Dopo una prima fase di pre-processing, volta alla preparazione del testo, abbiamo applicato diverse tecniche di classificazione, in modo da poter confrontare i risultati ottenuti e valutare quale tecnica si adatta meglio al nostro bisogno informativo. Pertanto abbiamo eseguito la classificazione tramite alcuni modelli di Machine Learning e Bert. In aggiunta abbiamo effettuato una Sentiment Analysis tramite Vader, un approccio basato sul lessico.

2 DATASET

Il progetto si basa sull'analisi del "Drug Review Dataset"[1], un insieme di dati ottenuto attraverso il *crowling* di alcuni siti di farmaceutica.
Il dataset riporta le recensioni riguardo a farmaci destinati all'uso umano ed è composto da 215063 osservazione e 6 attributi:

- "drugName" indica il nome del farmaco;
- "condition" indica il disturbo per cui il paziente assume il farmaco;
- "review" riporta la recensione del paziente riguardo al farmaco;
- "rating" indica la soddisfazione generale del paziente; il punteggio assume valori interi tra 0 e 10;
- "date" indica la data di rilascio della recensione;
- "usefulCount" indica il numero di utenti che hanno trovato utile la recensione.

3 PRE-PROCESSING

La fase di pre-processing è una fase fondamentale per ogni compito di *text mining*, poiché semplifica il contenuto del documento al fine di poterlo processare.

3.1 Tokenizzazione

La tokenizzazione consiste nel dividere il testo in unità (*token*) dense di significato, di solito tramite metodi statistici ed espressioni regolari.

3.2 Normalizzazione

È il processo che consiste nell'uniformare le parole, per portarle ad un unico formato. Nello specifico abbiamo:

- trasformato le lettere maiuscole in minuscole
- rimosso i numeri in quanto non sono d'interesse per l'analisi
- rimosso il carattere speciale "'" presente in qualche recensione
- rimosso gli spazi bianchi

3.3 Stemming

Lo stemming consiste in un processo grezzo che taglia le parole e le riconduce alla loro radice, rimuovendo affissi e desinenze. Lo svantaggio, rispetto alla lemmatizzazione, è che non riesce a disambiguare tra diverse forme base, con la conseguenza che si rischia di perdere il preciso significato delle parole. Per la lingua inglese ci sono diversi algoritmi di stemming, noi abbiamo utilizzato l'algoritmo *Snowball*, che viene considerato come la versione migliorata dell'algoritmo di Porter, in quanto risolve alcune questioni proprie dell'algoritmo.

3.4 Rimozione *stopwords*

Il processo consiste nell'eliminazione delle parole con bassa capacità discriminante da un testo (ad esempio articoli, congiunzioni, avverbi) oppure delle parole ad alta frequenza; in sostanza vengono rimosse le parole con poco contenuto semantico o parole molto ricorrenti. Un modo per determinare la lista delle *stopwords* consiste nell'ordinare le parole di un testo in base alla frequenza decrescente e considerare le parole più frequenti come *stopwords*.

Figure 1: Parole maggiormente ricorrenti

4 TEXT REPRESENTATION

Nella rappresentazione testuale si sceglie un modello per rappresentare il testo. È necessario infatti convertire il testo in forma numerica prima di applicare un algoritmo di *Machine Learning*.
Nello specifico abbiamo scelto di utilizzare il modello Bag-of-Words e lo schema di pesi TF-IDF.

4.1 Modello Bag-of-Words

L'idea sottostante al modello Bag-of-Words (BoW) è che un testo possa essere rappresentato per mezzo delle parole che contiene.
Per costruire il modello BoW abbiamo usato la classe *CountVectorizer* da *scikitlearn*, che prende in input un array di dati testuali e costruisce il modello.
Il modello BoW produce dei vettori che riportano la frequenza con cui la parola ricorre nel documento; poiché le parole di ogni singola recensione rappresentano un piccolo sottoinsieme di tutte le parole contenute nel vocabolario analizzato, la matrice prodotta è grande e sparsa (presenza di molti zeri).

4.2 Matrice TF-IDF

L'idea alla base dello schema di pesi TF-IDF è che non tutte le parole in un testo descrivono il contenuto con la stessa informatività. Quest'idea trova le basi nel lavoro di Luhn (Figura 1), che, riprendendo la legge di Zipf, ha definito la base per il calcolo dei pesi da assegnare alle parole di un documento. Nello schema TF-IDF vengono combinate due euristiche:

1. TF: la frequenza del termine, cioè il numero di volte che il termine t appare nel documento d, normalizzata rispetto al numero più elevato delle occorrenze;
2. IDF: il logaritmo dell'inversa del numero di documenti d in tutta la collezione in cui appare il termine t.

Lo schema TF-IDF permette di assegnare un peso alto alle parole comuni nel documento ed un peso basso alle parole rare.

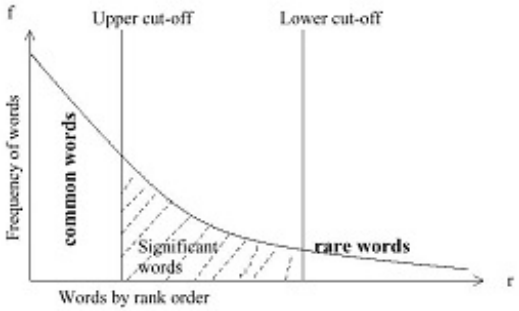

Figure 2: Analisi di Luhn

5 SENTIMENT ANALYSIS

5.1 Vader

Vader è uno strumento lessicale di *sentiment analysis* basato su regole (rule-based); utilizza una lista di *features* lessicali, come ad esempio parole, che vengono etichettate in base al loro orientamento semantico (positivo o negativo). Ha il vantaggio che, oltre a dire se un sentimento è positivo o negativo, dice anche quanto positivo o negativo sia, quindi Vader coglie sia la polarità che l'intensità del sentimento.
Il risultato dell'applicazione di Vader consiste in quattro elementi di sentiment:

- la polarità positiva
- la polarità negativa
- la posizione neutra
- il punteggio composto (*compound*), che è la somma di tutti i punteggi lessicali precedenti normalizzati tra -1 ed 1. Il valore di compound è quello che ci interessa, ed è la normalizzazione del punteggio di una frase calcolato come somma dei sentimenti presenti nel vocabolario, tenendo presente la deviazione standard.

Per l'implementazione di Vader abbiamo scelto di mantenere le *stopword* (ad esempio "but" o altre congiunzioni), perché forniscono indicazioni importanti per la classificazione, possono infatti spostare la polarità di un sentimento.

6 TEXT CLASSIFICATION

La classificazione testuale prevede che il testo venga assegnato ad una categoria predefinita, in questo caso la categoria "recensione positiva" o "recensione negativa". Si tratta di un apprendimento supervisionato: il classificatore viene addestrato sulla base degli esempi contenuti nel training set e successivamente viene valutato sul test set.

6.1 Moldelli di Machine Learning

I modelli di machine learning addestrati per eseguire la classificazione del testo in "recensione positiva" o "recensione negativa" sono:

- Support Vector Machine (SVM);
- Logistic Regression (LR);
- Random Forest;
- Decision Tree;
- XGBoost;
- Gaussian Naive Bayes;
- KNeighbors.

Per incrementare l'efficienza e ridurre il tempo computazionale dei modelli sopracitati, data la significativa dimensione delle matrici Bag-of-Words e TF-IDF, si è deciso di attuare due tecniche di riduzione della dimensionalità:

- kBest: un algoritmo di selezione delle migliori *features* basato, in questo caso, sull'ANOVA f-value con un p-value >= 0.7;
- PCA: è una procedura statistica che utilizza una trasformazione ortogonale per convertire un insieme di osservazioni di variabili eventualmente correlate in un insieme di valori di variabili linearmente non correlate, chiamate componenti principali [2]. La PCA ha permesso di ridurre ulteriormente il numero di *features*, mantenendo oltre il 95% della varianza totale dei dati.

Nello specifico, i modelli di classificazione utilizzati fanno riferimento a 4 categorie distinte:

- Euristica: L'obiettivo di un'euristica è produrre una soluzione in un lasso di tempo ragionevole che sia sufficientemente buona per risolvere il problema in questione[3]. I modelli euristici utilizzati sono: Decision Tree, Random Forest, KNeighbors e XGBoost;
- Separazione: questi algoritmi tentano di mappare i dati in uno spazio più ampio, con l'obiettivo di trovare l'iperpiano che meglio separa le variabili in base alla variabile di interesse utilizzando le funzioni. Il modello di separazione utilizzato è la SVM;
- Probabilistico: Nell'apprendimento automatico , un classificatore probabilistico è un classificatore in grado di prevedere, data un'osservazione di un input, una distribuzione di probabilità su un insieme di classi, piuttosto che produrre solo la classe più probabile a cui dovrebbe appartenere l'osservazione [4]. Il modello probabilistico utilizzato è il Gaussian Naive Bayes;
- Regressione: sono algoritmi molto flessibili e di facile comprensione in quanto è possibile misurare l'effetto delle diverse variabili nella classificazione grazie ai coefficienti assegnati dal modello. Il modello utilizzato è la Regressione Logistica.

6.2 BERT

BERT utilizza Transformer, un meccanismo *attention-based* che apprende le relazioni contestuali tra le parole (o le sotto parole) in un testo. Nella sua forma base, Transformer include due meccanismi separati:

- un codificatore che legge l'input di testo;
- un decodificatore che produce una previsione per l'attività.

Poiché l'obiettivo del BERT è generare un modello linguistico, 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 da destra a sinistra), il codificatore Transformer legge l'intera sequenza di parole contemporaneamente, pertanto è da considerarsi un modello bidirezionale. Questa caratteristica consente al modello di apprendere il contesto di una parola in base a tutto ciò che la circonda (a sinistra e a destra della parola)[5].

7 VALUTAZIONI

7.0.1 Vader

Applicando Vader alle recensioni abbiamo ottenuto un'accuratezza pari a 0.63; può essere considerato un risultato abbastanza buono.

7.0.2 Modelli di Machine Learning

I modelli sono stati valutati secondo il miglior punteggio ottenuto in termini di accuracy.

Per quanto riguarda l'approccio con la rappresentazione Bag-of-Words si sono ottenuti i seguenti risultati:

- Modelli Euristici: tra i modelli euristici il miglior risultato è stato raggiunto con il modello XGboost la cui accuracy si attesta ad un valore di 0.77.
- Modello di Separazione: il modello SVM ha raggiunto un accuracy pari a 0.78.
- Modelli Probabilistici: il modello Naive Bayes ha raggiunto un accuracy pari a 0.55.
- Modelli di Regressione: la Regressione Logistica, così come XGBoost, raggiunge un'accuratezza pari a 0.77.

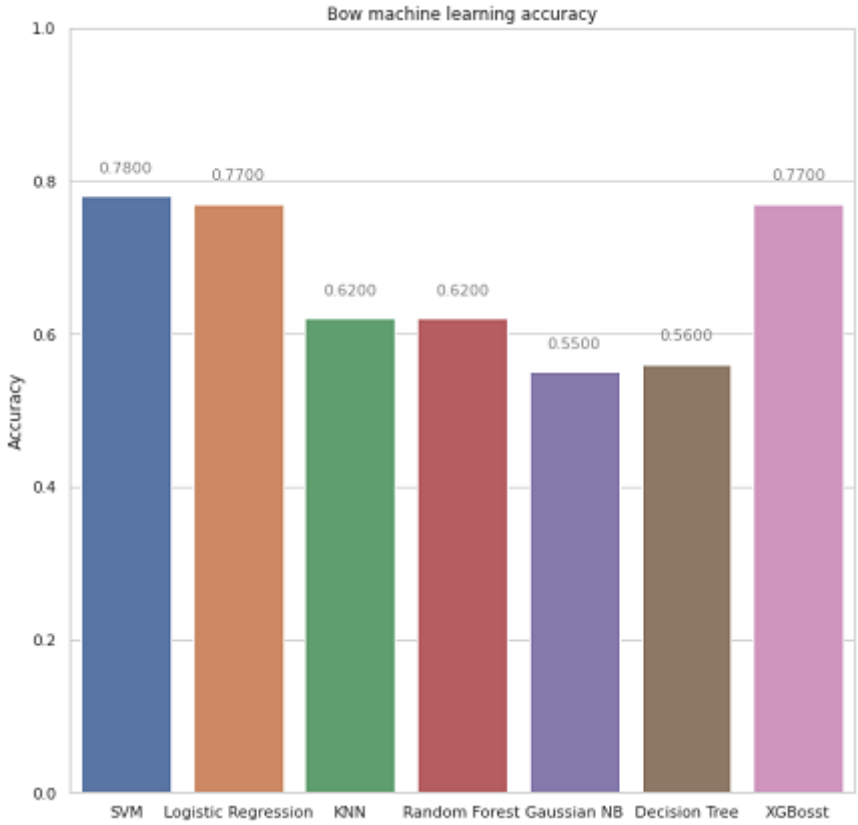

Figure 3: L'addestramento dei modelli è stato eseguito attraverso i parametri standard della libreria scikit-learn di Python

Per quanto riguarda l'approccio con la rappresentazione TF-IDF si sono ottenuti i seguenti risultati:

- Modelli Euristici: anche in questo caso, tra i modelli euristici, il modello XGBoost si è rivelato essere il miglior modello raggiungendo un'accuratezza pari al 77%.
- Modello di Separazione: Il modello SVM è risultato essere il miglior modello di Machine Learning con l'approccio TF-IDF raggiungendo un'accuratezza pari a 78%;
- Modelli Probabilistici:così come per l'approccio con la Bag-of-Words, il modello Naive Bayes ha ottenuto risultati scadenti, raggiungendo un'accuratezza del 55%;
- Modelli di Regressione: come in precedenza, anche con l'approccio con TF-IDF il modello di Regressione Logistica ha ottenuto buonissimi risultati, raggiungendo un'accuratezza del 77%

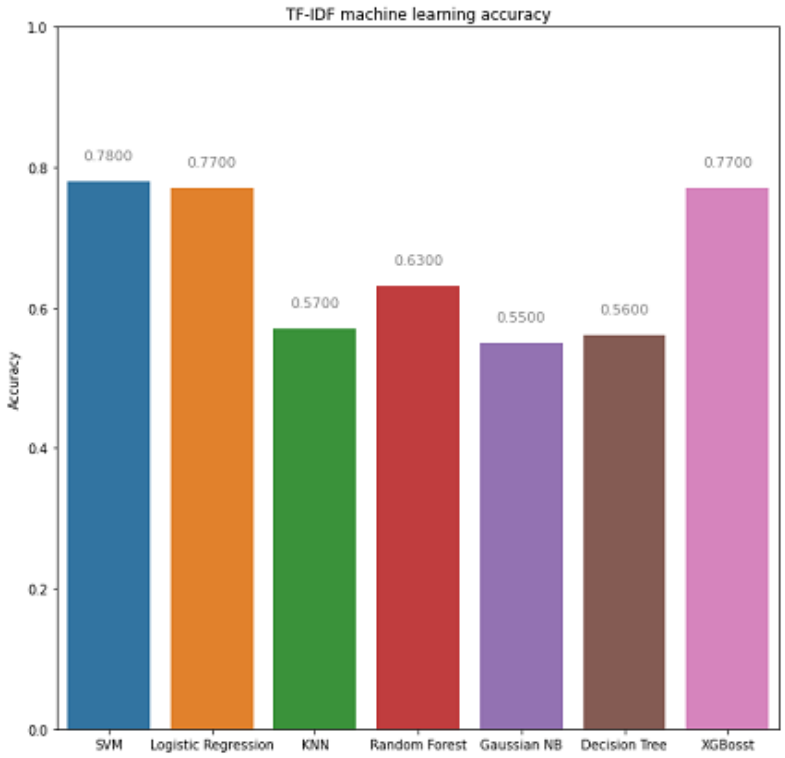

Figure 4: Anche in questo caso l'addestramento dei modelli è stato eseguito attraverso i parametri standard della libreria scikit-learn di Python

7.0.3 BERT

Contrariamente sia all'approccio con la Bag-of-Words, sia all'approccio con la TF-IDF, l'analisi effettuata con il modello BERT prende in considerazione 3 classi: recensione positiva, recensione negativa e recensione neutrale. L'accuratezza raggiunta con il modello BERT è pari al 70%.

	precision	recall	f1-score	support
positive	0.82	0.74	0.78	265
neutral	0.71	0.67	0.69	265
negative	0.59	0.68	0.63	265
accuracy			0.70	795
macro avg	0.71	0.70	0.70	795
weighted avg	0.71	0.70	0.70	795

Figure 5: Matrice di classificazione del modello BERT.

8 CONCLUSIONI

Tutti gli approcci implementati hanno restituito risultati soddisfacenti. Per quanto concerne i modelli di Machine Learning, il miglior risultato in termini di accuratezza si è raggiunto con il metodo supervisionato SVM.
Ovviamente non risulta corretto fare un confronto tra i tre metodi implementati, poiché si basano su fondamenti teorici diversi.

9 REFERENZE

1. Datset: https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%29
2. PCA: https://en.wikipedia.org/wiki/Principal_component_analysis#Intuition
3. Euristica: https://en.wikipedia.org/wiki/Heuristic_(computer_science)
4. Modello probabilistico: https://en.wikipedia.org/wiki/Probabilistic_classification
5. Bert: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270