

*Progetto per l’esame di* ***Text Mining and Search***

*Corso di Laurea in Data Science (a.a 2018/2019)*

Lavoro a cura di:

*Luca Gabellini (777786)*

*Matteo Provasi (782922)*

*Pierluigi Tagliabue (835211)*

# ANALISI RECENSIONI AMAZON

Sommario

[ANALISI RECENSIONI AMAZON 2](#_Toc535422690)

[DATI 3](#_Toc535422691)

[PREPROCESSING 3](#_Toc535422692)

[TOKENIZZAZIONE 3](#_Toc535422693)

[NORMALIZZAZIONE 4](#_Toc535422694)

[LEMMIZZAZIONE 4](#_Toc535422695)

[STOPWORDS 4](#_Toc535422696)

[TRAIN-TEST SPLIT 5](#_Toc535422697)

[MODELLI 5](#_Toc535422698)

[NAIVE BAYES 5](#_Toc535422699)

[RANDOM FOREST 6](#_Toc535422700)

[MODELLO LOGISTICO 6](#_Toc535422701)

[RISULTATI E CONCLUSIONI 6](#_Toc535422702)

[MUSICAL INSTRUMENTS 6](#_Toc535422703)

[RILEVANZA DEI TERMINI CONDIZIONATA PER CLASSE 7](#_Toc535422704)

[NUOVI MODELLI 7](#_Toc535422705)

[FEATURES 8](#_Toc535422706)

## DATI

I dati sono stati scaricati da uno dei siti consigliati dai docenti[[1]](#footnote-1). La repository presenta un elenco di file che contengono recensioni Amazon scritte in un arco temporale di 18 anni (fino a marzo 2013), divise per categorie di prodotto.

Il task scelto per il progetto è quello della text classification: il nostro obiettivo sarà quello di prevedere la categoria (di oggetti) di cui parla la recensione, per poi valutare la bontà di classificazione facendo un confronto tra categorie reali e previste.

Sono stati selezionati 9 dataset di recensioni relative ad altrettante categorie di oggetti:

* Baby products (prodotti per bambini)
* Cell phone and accessories
* Digital music
* Musical instruments
* Office products
* Pet supplies
* Tools and home improvements
* Toys and games
* Video games

I file sono stati caricati uno per volta su Python, effettuando una piccola correzione al formato in modo da poter leggere correttamente il file .json. Le informazioni contenute all’interno dei .json sono state poi memorizzate e unite in un dataset *pandas*. Data l’elevata numerosità delle recensioni di partenza, si è scelto di caricare il 20% delle osservazioni di ogni file, estratte in maniera casuale senza reinserimento. Il file finale comprende oltre 200 mila recensioni, una numerosità sufficiente per svolgere un corretto task di classificazione.

I file .json iniziali comprendevano differenti campi, quali il nome dell’autore o la data di pubblicazione. L’unica colonna di nostro interesse è risultata **textReview** (testo della recensione); è stata creata poi la colonna **category** (oggetto di cui parla la recensione), che verrà usata come target nei successivi modelli di classificazione. L’altra variabile testuale che si sarebbe potuta tenere in considerazione era il titolo della recensione, ma dopo un’attenta analisi si è ritenuto non necessario inserirla in quanto i titoli erano spesso composti da poche parole, che sarebbero potute diminuire ancora dopo il preprocessing, e si sarebbe rischiato di ottenere titoli senza potere discriminatorio.

## PREPROCESSING

Durante questo step sono state implementati i processi di tokenizzazione, normalizzazione, lemmizzazione e rimozione delle stopwords per standardizzare il testo, necessari per fornire un valido input alla successiva fase di classificazione.

### TOKENIZZAZIONE

Questo step riguarda la divisione del testo in singole parole, token. Data la struttura dei nostri dati, memorizzati in un dataset, è risultato più comodo non effettuare la tokenizzazione separatamente, ma includerla in ognuno degli step di preprocessing. Nello specifico:

* viene inizialmente implementata una list comprehension che passa in rassegna le recensioni del dataset
* la recensione viene divisa in singole parole utilizzando lo spazio come carattere di split
* si applica la funzione specifica del preprocessing sui token ottenuti
* le parole vengono ricongiunte fra di loro a ricreare la review, mantenendo inalterata la struttura del dataset.

### NORMALIZZAZIONE

Questo step ha il compito di riportare il testo in una forma standard in termini di caratteri. In questo passaggio sono state effettuate due operazioni principali:

* Lower case: Python, essendo un software case sensitive, considera caratteri minuscoli e maiuscoli come differenti; dunque due parole uguali, una scritta con la maiuscola, una con la minuscola, sono considerati come due token differenti. Riportare tutte le parole in minuscolo può comportare dei problemi in quanto esistono parole che sono scritte in maiuscolo perché sono delle sigle o simili. Non sono stati fatti dei passaggi intermedi in quanto per il nostro compito di classificazione, l’influenza che può avere la trasformazione in minuscolo di una parola che dovrebbe essere maiuscola è trascurabile. Avrebbe avuto più senso fare degli accorgimenti nel caso si sarebbe dovuto fare una entity recognition.
* Caratteri speciali: tutti i caratteri speciali e i segni di punteggiatura sono stati sostituiti con uno spazio vuoto. In un primo momento questi caratteri non venivano sostituiti ma eliminati direttamente, tuttavia ci si è accorti in seguito che alcuni utenti nelle loro recensioni non mettevano lo spazio fra il segno di punteggiatura e la parola successiva. In questa maniera l’eliminazione del carattere portava all’unione di due parole in un token privo di significato, da qui la necessità di introdurre uno spazio come elemento sostitutivo.

### LEMMIZZAZIONE

Questo processo ha il compito di riportare le parole alla loro forma base, cancellando tutte le inflessioni date dal discorso. La lemmizzazione è stata effettuata per verbi, sostantivi ed aggettivi. Per i verbi sono rimosse le forme relative alla terza persona singolare le forme infinite, i nomi sono portati alla versione singolare, mentre per gli aggettivi è rimosso un eventuale comparativo.

### STOPWORDS

Una lista contenente le principali stopwords è stata ottenuta dalla libreria *ntlk*[[2]](#footnote-2) (Natural Language Toolkit). Dalla lista originaria sono poi stati inseriti dei termini che risultavano essere chiaramente delle stopwords, ma per via dei processi di normalizzazione e lemmizzazione precedenti avevano assunto una forma particolare che non le faceva rientrare nella lista di *nltk* (ad esempio il termine *haven’t* era stato trasformato in *havent* e non era riconosciuto dal processo di stemming). Estesa la lista si è proceduto con la rimozione delle parole che avevano una lunghezza inferiore ai tre caratteri in quanto erano principalmente numeri o gruppi di due caratteri che per errori di battitura erano diventati dei token senza alcun significato che avrebbero creato dei problemi ai modelli di classificazione. Avendo a che fare con dataset riguardati recensioni provenienti da ambiti differenti, non si è utilizzato nessuna lista ad hoc per le stopwords, ma si è rimasti su quella di base ampliata opportunamente.

*Esempio di text preprocessing su una recensione del dataset:*

Documento originale:

GREAT light so far....after 8 months it is going strong ...i hope it lasts....i can believe it has a motion sensor....!!

1. Testo convertito a lower case:

great light so far....after 8 months it is going strong ...i hope it lasts....i can believe it has a motion sensor....!!

1. Rimozione Punteggiatura:

great light so far after 8 months it is going strong i hope it lasts i can believe it has a motion sensor

1. Testo lemmizzato:

great light so far after 8 month it be go strong i hope it last i can

believe it have a motion sensor

1. Rimozione Stopwords:

great light far month strong hope last believe motion sensor

### TRAIN-TEST SPLIT

Una volta completata la fase di text preprocessing, il dataset è diviso in train e test (proporzioni di default con 75% di osservazioni nel train e 25% nel test), optando per una stratificazione proporzionale alla distribuzione delle categorie.

Si sceglie di rappresentare i documenti come vettori, con pesi calcolati con la procedura TF-IDF. Per fare ciò si ricorre alla funzione *TfidfVectorizer* del pacchetto *sklearn[[3]](#footnote-3).* L’algoritmo viene addestrato sul train, e restituisce una rappresentazione dei dati di interesse in forma matriciale (document-term matrix). L’algoritmo sceglie le features (termini) più rilevanti tra le recensioni, che andranno a costituire le colonne della matrice sparsa. Le righe saranno di numerosità pari al numero di documenti contenenti nel train. Il numero massimo di features (termini) viene fissato a 5000. Questa quantità è stata scelta in quanto inserendo più feature sarebbero stati selezionati termini con poco potere discriminatorio o scorretti da un punto di vista sintattico. Infine anche il dataset di test viene convertito in forma matriciale.

La rappresentazione delle recensioni in formato vettoriale è risultata ideale come input per la successiva fase di classificazione.

## MODELLI

Per il task di classificazione sono stati implementati vari modelli di machine learning. Ognuno di questi modelli viene addestrato sui dati di training, dove apprende quali siano le relazioni che legano le recensioni alla loro classe. Una volta concluso questo step il classificatore è applicato ai dati di test, restituendo come output la classe di appartenenza più verosimile per ciascuna recensione.

### NAIVE BAYES

Il metodo utilizza il teorema di Bayes per stimare la probabilità di appartenenza di un documento ad una classe. Nella fase di training si calcolano le probabilità condizionate delle classi rispetto ai documenti; applicando il teorema di Bayes la classe del documento prevista è quella che massimizza la probabilità a posteriori di appartenenza C[[4]](#footnote-4):

Il metodo presuppone l’ipotesi di indipendenza fra i termini all’interno del documento e che l’ordine in cui compaiono non sia rilevante. Grazie alla prima ipotesi è possibile scrivere le probabilità condizionate come una produttoria. Viene fatta una correzione di Laplace per ovviare alla presenza di nuove parole nel dataset di test, la cui probabilità condizionata sarebbe nulla.

### RANDOM FOREST

La random forest è un meta-stimatore che addestra un numero predefinito di alberi casuali su differenti campioni del dataset, mediando i risultati per migliorare l’accuracy e controllare l’overfitting. Sono stati creati più modelli utilizzando un numero diverso di alberi. Per ogni albero si sceglie un subset casuale delle features di partenza: in genere se si hanno features, ogni albero deve essere creato usandone al massimo (nel nostro caso). La procedura di split viene governata utilizzando l’indice di gini: i nodi figli presenteranno una distribuzione che minimizzi l’indice, ovvero massimizzi la purezza intra-nodo. Procedendo in maniera iterativa si creano diversi alberi, una volta giunti all’ultimo le recensioni sono classificate in base alla classe a cui sono state assegnate con maggior frequenza da tutti gli alberi.

I risultati migliori sono stati ottenuti nel modello a 100 alberi; da qui in avanti si parlerà semplicemente di Random Forest sottintendendo che si tratti del modello con 100 alberi

### MODELLO LOGISTICO

Viene infine proposto un modello logistico con parametri settati in modo tale da rendere possibile una classificazione multinomiale: come solver si opta per il ‘sag’, ideale per dataset di grosse dimensioni come il nostro. Selezionando ‘multinomial’ come metodo di classificazione viene utilizzata una funzione di loss (detta softmax) che calcola la probabilità che la data recensione appartenga ad ognuna delle possibili 9 classi. La recensione verrà assegnata alla classe a cui è associata la probabilità più elevata.

## RISULTATI

### MUSICAL INSTRUMENTS

Le recensioni appartenenti a questa classe sono molto minori in numerosità rispetto a quelle delle altre categorie. Analizzando gli score relativi alla bontà di classificazione la precision per musical instruments risulta molto alta, mentre la recall risulta molto inferiore rispetto a quella di altre classi.

La soluzione trovata per sistemare questo problema è quella di andare a considerare l’intero dataset relativo alla classe musical instruments. Considerando il file completo si ha un totale di poco più di 10 mila osservazioni, numerosità paragonabile a quella della seconda classe meno numerosa. Per importare i nuovi dati si è creato un nuovo dataset al quale si sono applicati tutti i passaggi di preprocessing. Da qui si è proceduto con un nuovo split del dataset fra train e test.

RILEVANZA DEI TERMINI CONDIZIONATA PER CLASSE

Come analisi esplorativa utile ai fini del nostro obiettivo sono stati individuati i termini più rilevanti e caratteristici di ogni categoria. Una misura popolare per calcolare la suddetta rilevanza è la statistica chi-quadro:

funzione delle feature *f* e del target *t*, dove:

* A rappresenta il numero di volte in cui *f* e *t* co-occorrono
* B rappresenta il numero di volte in cui *f* occorre senza *t*
* C rappresenta il numero di volte in cui *t* occorre senza *f*
* D rappresenta il numero di volte in cui né *f* né *t* occorrono.

Bisogna specificare che la statistica è stata calcolata utilizzando i pesi TF-IDF, che possono essere visti come delle frequenze normalizzate/pesate.

# Baby:

1. baby

2. diaper

3. bottle

# Cell Phones and

Accessories:

1. phone

2. case

3. charge

# Digital Music:

1. album

2. song

3. quot

# Musical Instruments:

1. guitar

2. string

3. pedal

# Office Products:

1. printer

2. ink

3. paper

# Pet Supplies:

1. dog

2. cat

3. treat

# Tools and Home

Improvement:

1. doll

2. toy

3. kid

# Toys and Games:

1. tool

2. light

3. bulb

# Video Games:

1. game

2. graphic

3. play

### RISULTATI SUL DATASET RICAMPIONATO

I risultati dei modelli riadattati sul dataset ricampionato presentano delle performance simili a quelle sul vecchio dataset: l’accuracy globale rimane pressochè invariata:

|  |  |  |
| --- | --- | --- |
|  | Dataset originale | Dataset ricampionato |
| **Modello** | **Accuracy globale** | |
| Naive Bayes | *0.883* | *0.883* |
| Random Forest | *0.864* | *0.864* |
| Logistic | *0.908* | *0.906* |

I miglioramenti si possono però notare nella recall relativa alla classe ricampionata, Musical Instruments, di cui sono stati aggiunti tutti i record a nostra disposizione:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Dataset originale | | Nuovo dataset | |
| **Modello** | **Precision M.I.** | **Recall M.I.** | **Precision M.I.** | **Recall M.I.** |
| Naive Bayes | *0.83* | *0.31* | *0.83* | *0.78* |
| Random Forest | *0.74* | *0.35* | *0.74* | *0.73* |
| Logistic | *0.86* | *0.62* | *0.83* | *0.75* |

### FEATURES

Fino a questo punto i modelli sono stati costruiti utilizzando un numero di feature fissato a 5000. Si sono valutate le performance del modello migliore al variare del numero di features considerate. Il modello migliore è risultato il Logistico applicato al secondo dataset, anche se presentava una minore accuracy, non produceva errori sistematici di classificazione per la classe *Musical Instruments*. Utilizzando un numero minore le performance erano peggiori, mentre miglioravano costantemente fino a 30’000 per poi diminuire. Il miglioramento ottenuto, sia in termini di accuratezza, precision e recall non è significativo (aumento di pochi decimali), quindi si può considerare corretto il numero di features scelte all’inizio.

## CONCLUSIONI

Le numerose librerie messe a disposizione dal software python hanno permesso espletare in maniera efficace il task di text classification preposto: tutti gli step implementati, partendo dalla data ingestion, passando per il text preprocessing per arrivare classificazione, hanno fornito ottimi risultati, raggiunti con tempi di elaborazione più che soddisfacenti data la grossa mole di dati processata. Si è rivelato interessante l’esito dei termini più rilevanti per ciascuna categoria, che ha permesso di dare una visione sintetica delle features più rilevanti. Le statistiche utili per valutare la bontà classificativa sono risultate molto elevate per tutti i modelli di machine learning addestrati.

Il lavoro fin qui fatto potrebbe essere riadattato e utilizzato come spunto per migliorare il reccomendation system di Amazon, sfruttando gli insight utili per comprendere le preferenze degli utenti per ogni categoria di oggetto considerata.

1. <https://snap.stanford.edu/data/web-Amazon.html> [↑](#footnote-ref-1)
2. <https://www.nltk.org/> [↑](#footnote-ref-2)
3. <https://scikit-learn.org/stable/> [↑](#footnote-ref-3)
4. <https://web.stanford.edu/class/cs124/lec/naivebayes.pdf> [↑](#footnote-ref-4)