UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA
Scuola di Scienze
Corso di Laurea Magistrale in Data Science

Progetto Text mining and search

Text classification e Topic Modeling - Goodread recensioni

Corso condotto da:
Prof.ssa Gabriella Pasi
Prof. Marco Viviani

Svolto da:
Claudio Fadda matr. 813499
Monica Vivace matr.

Anno Accademico 2021-2022

Indice

Capitolo 1

Introduzione

Goodreads è un social network lanciato nel 2007 focalizzato sui libri. Gli utenti registrati possono aggiungere alla loro libreria virtuale dei libri e condividere commenti, scrivere delle recensioni, esprimere dei voti, creare e partecipare a delle discussioni.

Questo progetto si pone l'obiettivo di svolgere un task di Text Classification e un task di Topic Modeling. In particolare, per svolgere la Text Classification verranno prese in considerazione come tecniche di rappresentazione TF-IDF e Word2vect e come modelli predittivi: Logistic Regression, Random Forest e Stochastic Gradient Descent. La valutazione delle performance dei modelli avverrà tramite diverse misure quali Accuracy, Recall, Precision. Per ciò che riguarda il Topic Modeling si prenderanno in considerazione due tecniche: Latent Semantic Analysis (LSA) e Latent Dirichlet allocation (LDA) valutandone le performance.

Capitolo 2

Dati

I dati contenuti nel dataset Goodreads sono stati raccolti nel 2017 tramite scraping del sito Goodreads. Questi dati sono stati archiviati in tre gruppi: *meta-data of the books*, *user-book interactions*, *users detailed book reviews*. Vista la grossa mole di dati, questi sono stati anche divisi per genere in dataset di medie dimensioni in modo da riuscire a manipolarli meglio.

Per i nostri task abbiamo deciso di andare a considerare i dataset *users detailed book reviews* relativi ai seguenti generi:

- **Children**;
- **Comics & Graphic**;
- **Poetry**.

. I dataset sono in formato JSON e presentano le seguenti colonne:

- **user_id**: Identificativo dell'utente;
- **book_id**: Identificativo del libro;
- **review_id**: Identificativo della recensione;
- **rating**: Valutazione del libro;
- **review_text**: Testo della recensione;
- **date_added**;
- **date_updated**;
- **read_at**;
- **started_at**;
- **n_votes**: Numero di voti per la recensione;
- **n_comments**: Numero di commenti per la recensione dell'utente.

Per il nostro obiettivo consideriamo i seguenti attributi:

- **book_id**;
- **review_id**;
- **review_text**.

2.1 Ulteriori considerazioni

Nonostante il maggior numero di recensioni sia scritto in lingua inglese, non è raro che alcune di esse siano scritte in una lingua differente. Per i task da noi considerati la lingua è un fattore importante, quindi decidiamo di considerare solamente le recensioni scritte in lingua inglese.

Infine, visto che dataset di generi diversi hanno un numero di recensioni molto differente tra di loro, una volta che sono state selezionate in base alla lingua, è stata presa la decisione di eseguire un campionamento casuale pari a 60.000 righe.

Capitolo 3

Pre-processing

Una delle fasi più importanti, specie se si lavora con dati testuali è data dalla *text-processing*, sono necessari infatti diversi passaggi prima di poter elaborare e manipolare il testo. In particolare risulta fondamentale fornire alla macchina una rappresentazione formale del testo in modo tale che il testo sia facilmente manipolabile, si parla a tal proposito di rappresentazione *Bag of Words*. A seconda dei task di text mining, le tecniche di pre-processing possono variare, in base al problema si valutano infatti quelle più adeguate. In particolare gli step fondamentali prevedono la tokenization, stop words removal, case folding e normalization. In aggiunta è possibile svolgere anche Stemming e Lemmatization.

3.1 Tokenization

Il processo di tokenization permette di suddividere il testo (considerato come una sequenza di caratteri) in una lista di sotto-sequenze, i token. Questi devono essere quanto più significativi (dopo ulteriori passaggi) in modo da essere rappresentativi del testo. Un token è dato da una parola, una sequenza di caratteri, oppure una sequenza di più termini; ciò che permette di identificare i token sono le regole del tokenizer che possono essere attuate attraverso le espressioni regolari oppure metodi statistici.

Esempio di tokenizzazione

Frase: *Il pollo arrosto piace a tutti* → Output: “Il”, “pollo”, “arrosto”, “piace”, “a”, “tutti”

3.2 Normalization

La normalizzazione consente di riportare i termini alla stessa forma. Termini diversi che hanno lo stesso significato vengono considerati come un unico termine. Questo passaggio è molto importante soprattutto con quei termini che assumono una forte variabilità per esempio le sigle oppure le date (possono essere scritte in forme diverse e anche in ordine differente).

Esempio di normalizzazione

“ok”, “okay”, “Ok” → “ok”.

3.3 Stemming

Il processo di stemming riduce la forma flessa della parola alla sua forma radice. Per la lingua inglese l’algoritmo più utilizzato è quello di *Porter*.

3.4 Lemmatization

Nel processo di lemmatization invece, si riduce la parola alla sua forma base, questo consente successivamente di aggregare insieme parole che presentano forma base simile.

3.5 Stopwords removal

Questo processo viene usato per eliminare tutte le parole più comuni nel testo che sono considerate poco significative; queste non sono universali ma variano in base alla lingua che si sta considerando.

Capitolo 4

Text Classification

Il primo obiettivo riguarda la classificazione delle recensioni. La Text Classification è una tecnica supervisionata di Machine Learning che consiste nell'assegnare una classe (dato un insieme di classi) a degli oggetti, nel caso del Text Mining gli oggetti considerati sono i documenti testuali.

Si vuole quindi sviluppare la seguente formulazione:

$$h : D \rightarrow C \tag{4.1}$$

Dove D è un insieme di documenti, C è un insieme finito di classi.

4.1 Fasi preliminari

Prima di procedere ad implementare i modelli, oltre alle fasi di processamento del testo classiche quali rimozione delle stop words, delle emoji, caratteri, ecc. sono state implementate ulteriori tecniche che consentissero di ottenere le sole parole significative e di migliorare le performance dei nostri modelli.

4.1.1 Tokenization e Stemming

In primo luogo è stata svolta la Tokenization, in modo da ottenere, dato un testo, i *token* e successivamente Stemming. L'utilizzo di tale tecnica consentirebbe di ridurre parole molto simili alla stessa radice, per esempio se trovassimo in una frase del training set diverse parole come "are" mentre nel test set trovassimo una frase con un sola parola di questo tipo "is", lo stemming ridurrebbe entrambe ad una stessa feature cioè "be". A livello computazionale ci sarebbe sicuramente un risparmio di costo, in quanto verrebbero considerate meno feature. Uno dei problemi che potrebbe derivare da questa tecnica è il fatto di non prendere in considerazione l'aspetto semantico e il contesto delle parole.

Considerando l'utilizzo di *reviews* esclusivamente in lingua inglese per lo step relativo allo stemming si è scelto di usare l'**algoritmo di Porter**.

4.1.2 Divisione in train e test set

Il dataset viene poi suddiso in train e test set, applicando la regola dell'$80-20$. Viene inoltre messa in atto una suddivisione randomica stratificata basata sulla colonna "genre" in modo tale che le recensioni relative ai tre generi siano ugualmente bilanciate all'interno dei dataset.

4.2 Rappresentazione dei documenti

Si sceglie di implementare i 3 modelli sopra citati su due rappresentazioni del testo differenti:

- Text-document TF-IDF (Term Frequency - Inverse Document Frequency) weighted representation;
- Word2Vec representation.

4.2.1 TF-IDF

Un modo per rappresentare il testo è attraverso il modello "bag of words", basato sul fatto di interpretare un documento come un vettore di numeri reali. La rappresentazione tramite TF-IDF (Term Frequency - Inverse Document Frequency) permette di andare a pesare questi vettori secondo il seguente schema:

- **Term frequency (tf)**: $tf_{t,d}$ rappresenta la frequenza di occorrenza del termine t nel documento d;
- **Inverse document frequency (idf)**: $idf_t = \log\left(\frac{D}{df_t}\right)$, rappresenta quanti documenti (della collezione) contengono il termine t.

La combinazione di questi due pesi, permette di ottenere lo schema $tf - idf$:

$$w_{t,d} = \left(\frac{tf_{t,d}}{max_{ti} tf_{ti,d}}\right) \cdot log_{10}\left(\frac{D}{df_t}\right) \tag{4.2}$$

Il peso quindi aumenta con il numero di occorrenze all'interno del documento e aumenta con la rarità del termine all'interno della collezione.

Nel nostro caso implementiamo sia per il training set sia per il test set TfidfVectorizer, che consente di rappresentare i singoli documenti sotto forma di Matrici Documento-Termine, in cui il peso di ogni termine all'interno del documento viene calcolato utilizzando il metodo TF-IDF.

4.2.2 Word2Vec

Word2vect include un insieme di modelli utilizzati per ottenere word embedding, quindi per rappresentare le parole del testo mediante vettori densi. E' una rete neurale artificiale che cattura le co-occorrenze rispetto al sottoinsieme dei dati della collezione di documenti. Produce risultati migliori tanto più alta è la dimensionalità e tanto più alta è la varietà delle word embeddings che rappresentano la semantica dei nostri termini. In particolare comprende due tecniche: Skip-gram e CBOW, queste differiscono dal fatto che Skip-gram inizia da una target word e poi va ad apprendere una o più context-word. Nel caso del CBOW parte da un insieme di context-words e va ad apprendere la rappresentazione relativa alle target-words.

L'idea alla base è minimizzare la seguente funzione obiettivo:

$$J = 1 - P(context|word) \leftarrow min \tag{4.3}$$

o viceversa massimizzare la probabilità di occorrenza fra le due parole.

Nel nostro caso implementiamo sia per il training set sia per il test set Word2Vec, in particolare si sceglie di rappresentare il documento con un vettore pari a 300.

4.3 Modelli di classificazione testati

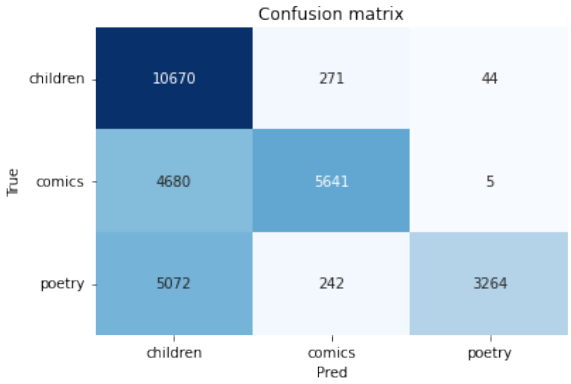
Come qualsiasi altro task di Text Mining, anche nella classificazione non esiste un singolo approccio capace di ottenere buoni risultati, per questo motivo sono stati considerati diversi algoritmi di classificazione che permettono di lavorare nell'ambito della Single-Label Multi Class. In particolare si è scelto di sviluppare:

- Random Forest;
- Regressione logistica;
- SGD (Stochastic Gradient Descent).

4.3.1 Comparazione dei risultati

TF-IDF

Di seguito vengono riportati i risultati ottenuti su tutti e tre i modelli .

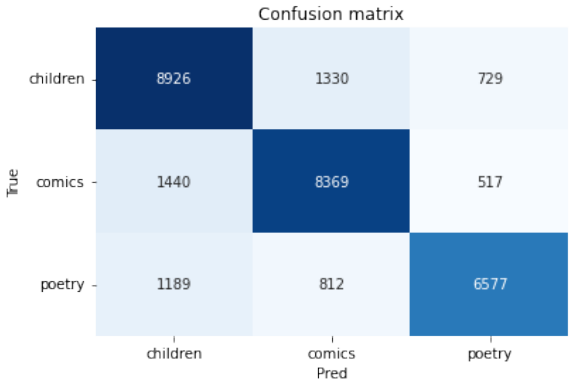



```
Train Accuracy        0.64932
Test Accuracy         0.65492
Test Precision        0.80811
Test Recall           0.63271
Test F1               0.63768
Test F2               0.61899
Classification report:
              precision    recall  f1-score   support

    children    0.52248   0.97132   0.67947     10985
      comics    0.91664   0.54629   0.68459     10326
      poetry    0.98521   0.38051   0.54899      8578

    accuracy                        0.65492     29889
   macro avg    0.80811   0.63271   0.63768     29889
weighted avg    0.79145   0.65492   0.64379     29889
```

Figura 4.1: Random Forest - Confusion Matrix e Risultati

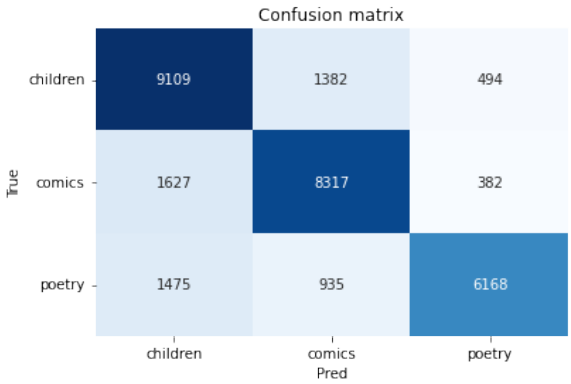



```
Train Accuracy        0.80070
Test Accuracy         0.79869
Test Precision        0.80314
Test Recall           0.79659
Test F1               0.79911
Test F2               0.79742
Classification report:
              precision    recall  f1-score   support

    children    0.77248   0.81256   0.79201     10985
      comics    0.79621   0.81048   0.80328     10326
      poetry    0.84073   0.76673   0.80202      8578

    accuracy                        0.79869     29889
   macro avg    0.80314   0.79659   0.79911     29889
weighted avg    0.80027   0.79869   0.79878     29889
```

Figura 4.2: Logistic Regression - Confusion Matrix e Risultati




```
Train Accuracy        0.78906
Test Accuracy         0.78939
Test Precision        0.80124
Test Recall           0.78457
Test F1               0.78955
Test F2               0.78583
Classification report:
              precision    recall  f1-score   support

    children    0.74597   0.82922   0.78539     10985
      comics    0.78211   0.80544   0.79361     10326
      poetry    0.87564   0.71905   0.78966      8578

    accuracy                        0.78939     29889
   macro avg    0.80124   0.78457   0.78955     29889
weighted avg    0.79567   0.78939   0.78945     29889
```

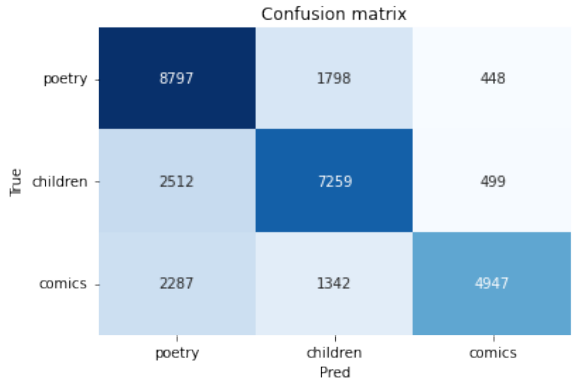
Figura 4.3: Stochastic Gradient Descent - Confusion Matrix e Risultati

Representation TF-IDF						
Tentativo	Modello	Accuracy	Class	Precision	Recall	f1 Score
1	Logistic Reg.	0.79	Children	0.77	0.81	0.79
			Comics	0.79	0.81	0.80
			Poetry	0.84	0.76	0.80
2	Random Forest	0.65	Children	0.52	0.97	0.67
			Comics	0.91	0.54	0.68
			Poetry	0.98	0.38	0.55
3	SGD	0.79	Children	0.74	0.83	0.78
			Comics	0.78	0.80	0.79
			Poetry	0.87	0.72	0.79

Tabella 4.1: Risultati modelli - rappresentazione TF - IDF

Dai risultati sembra che la Regressione Logistica e lo Stochastic Gradient Descent forniscano risultati molto simili, mentre il Random Forest è poco performante. Si nota inoltre che tutti e tre i modelli classificano molto bene il genere Children, seguito poi da Comics e Poetry. Infine in termini di efficienza, il modello Logistic Regression, richiede un costo computazionale inferiore rispetto al SGD. Quindi, se dovessimo fare una scelta, a parità di risultati in termini di efficacia si sceglierebbe la Regressione Logistica.

4.3.2 Word2vec

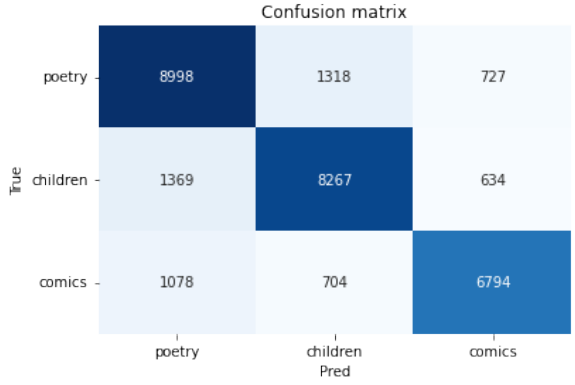



```
Train Accuracy      0.70274
Test Accuracy       0.70270
Test Precision      0.72813
Test Recall         0.69342
Test F1             0.70008
Test F2             0.69393
Classification report:
              precision    recall  f1-score   support

    children    0.64703   0.79661   0.71407     11043
      comics    0.69805   0.70682   0.70240     10270
      poetry    0.83933   0.57684   0.68376      8576

    accuracy                        0.70270     29889
   macro avg    0.72813   0.69342   0.70008     29889
weighted avg    0.71974   0.70270   0.70137     29889
```

Figura 4.4: Random Forest - Confusion Matrix e Risultati

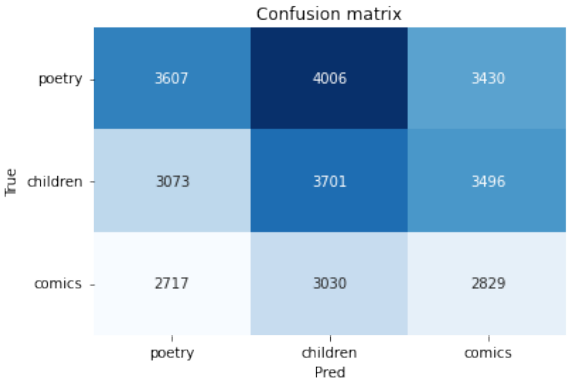



```
Train Accuracy      0.80557
Test Accuracy       0.80494
Test Precision      0.80759
Test Recall         0.80400
Test F1             0.80554
Test F2             0.80455
Classification report:
              precision    recall  f1-score   support

    children    0.78619   0.81481   0.80025     11043
      comics    0.80348   0.80497   0.80422     10270
      poetry    0.83311   0.79221   0.81215      8576

    accuracy                        0.80494     29889
   macro avg    0.80759   0.80400   0.80554     29889
weighted avg    0.80559   0.80494   0.80503     29889
```

Figura 4.5: Logistic Regression - Confusion Matrix e Risultati




```
Train Accuracy       0.33845
Test Accuracy        0.33915
Test Precision       0.33952
Test Recall          0.33896
Test F1              0.33798
Test F2              0.33828
Classification report:
              precision    recall  f1-score   support

    children    0.38385   0.32663   0.35294     11043
      comics    0.34470   0.36037   0.35236     10270
      poetry    0.29001   0.32987   0.30866      8576

    accuracy                        0.33915     29889
   macro avg    0.33952   0.33896   0.33798     29889
weighted avg    0.34347   0.33915   0.34003     29889
```

Figura 4.6: Stochastic Gradient Descent - Confusion Matrix e Risultati

Representation Word2vec						
Tentativo	Modello	Accuracy	Class	Precision	Recall	f1 Score
1	Logistic Reg.	0.80	Children	0.78	0.81	0.80
			Comics	0.80	0.80	0.80
			Poetry	0.83	0.79	0.81
2	Random Forest	0.70	Children	0.64	0.79	0.71
			Comics	0.70	0.70	0.70
			Poetry	0.84	0.57	0.68
3	SGD	0.34	Children	0.38	0.32	0.35
			Comics	0.34	0.36	0.35
			Poetry	0.29	0.33	0.31

Tabella 4.2: Risultati modelli - rappresentazione Word2vec

In questo caso dai risultati sembra che la Regressione Logistica e il Random Forest forniscano i risultati migliori, mentre l' SGD è molto poco performante. Nei primi due modelli vediamo che rispetto al caso precedente (TF-IDF) il genere Poetry è quello che ottiene buone performance.

Anche in questo caso la Logistic Regression, richiede un costo computazionale inferiore rispetto al SGD che invece risulta molto più oneroso in termini computazionali e poco efficace. Quindi, se dovessimo fare una scelta, a parità di risultati in termini di efficacia si sceglierebbe anche in questo caso la Regressione Logistica.

Capitolo 5

Topic modeling

Il Topic Modeling è una tecnica non supervisionata che ha l'obiettivo di trovare un insieme di topic attraverso la scansione di documenti testuali, ricercando parole e pattern al loro interno; questi cluster di parole che si andranno a formare saranno una rappresentazione dei topic.

5.1 Fasi preliminari

Dopo aver eseguito il pre-processing iniziale, eseguiamo il task di Lemmatization in modo da portare le parole alla loro forma base; questo passaggio è utile per consentire di individuare i topics in modo più semplice in quanto anche in questo caso si riduce il numero di feature da considerare. Come già sopra esposto relativamente allo Stemming, anche la Lemmatization, riportando le parole alla loro forma base consente di definire una sola feature per più parole con forma base uguale. Un aspetto negativo riguarda il fatto che non viene tenuto in considerazione il contesto, quindi parole che possono essere ricondotte alla stessa forma base potrebbero in realtà fare riferimento a due contesti e completamente diversi.

5.2 Rappresentazione dei documenti

Si implementano le due tecniche principali per il Topic Modeling:

- Latent Semantic Analysis (LSA);
- Latent Dirichlet Allocation (LDA).

5.3 LSA

La Latent Semantic Analysis è uno dei metodi più conosciuti per svolgere Topic Modeling. L'LSA prende in input la matrice Documento-Termine e la decompone in due matrici separate: una matrice Documento-Topic e una matrice Topic-Termine, tale relazione consente di mettere in relazione i documenti ai topic e i topic ai termini.

Basandosi sull'ipotesi distribuzionale:

- La semantica delle parole può essere individuata in base al contesto e quindi alle parole che circondano la parola target.
- Sotto l'ipotesi distribuzionale quindi, la semantica di due parole sarà simile se tendono a co-occorrere in un contesto simile.

L'LSA calcola quanto frequentemente le parole occorrono nei documenti, per questo motivo sia la conoscenza della sintassi (ordine delle parole) che quella della semantica (diversi significati di una

stessa parola) viene meno. Un modo classico per calcolare le frequenze è usare TF-IDF, in modo da poter calcolare la matrice Documento-Termine che contiene i valori associati ad ogni termine per un dato documento.

Per implementare questa tecnica è stato utilizzato TruncatedSVD.

5.4 LDA

Il Latent Dirichlet Allocation rappresenta un'ulteriore tecnica di Topic Modeling. Svolge una categorizzazione dei documenti mediante i topic, tramite una distribuzione delle parole. L'idea generale alla base è:

- Le parole sono generate dai topic;
- Ogni documento ha una data probabilità di usare particolari topic per generare una parola;
- Si cerca di trovare quali topic dati i documenti vengono utilizzati per generare le parole.

I principali passi che l'LDA compie sono:

1. Si parte con M documenti e si sceglie il numero k di topic che si vuole andare a ricercare all'interno dei testi;
2. Il modello, e gli M documenti sono espressi come una combinazione dei k topic;
3. L'algoritmo trova il peso delle connessioni tra documenti-topic e tra topic-parole;
4. Infine l'algoritmo restituisce i diversi topic ed ad ognuno di essi sono associate le parole con la probabilità di verificarsi all'interno di quel topic.

L'algoritmo che implementa l'LDA topic model è diverso rispetto all'algoritmo che implementa LSA poiché quest'ultimo necessita in input una rappresentazione Count Based. L'algoritmo utilizzato è il seguente CountVectorizer.

5.5 Modelli di classificazione testati

Word Cloud

Riportiamo la seguente immagine, definita altrimenti "word cloud" che ci permette di avere una idea delle parole che compaiono più spesso all'interno dei documenti testuali.

Figura 5.1: Word Cloud

5.5.1 Comparazione dei risultati

LSA

Il metodo LSA restituisce i seguenti risultati:

- **Topic 0**: one, like, love, poem, really, time, good, great, poetry, much;
- **Topic 1**: poem, poetry, collection, poet, beautiful, word, love, favorite, work, line;
- **Topic 2**: love, illustration, cute, kid, great, beautiful, fun, child, series, picture.

Osservando i topic restituiti, si nota come il metodo LSA non riesca a individuare i topic principali delle recensioni relative alle classi di libri. Vediamo che sia nel topic 0 che nel topic 1, emergono topic simili che non consentono di discriminare le recensioni (la parola "poetry" per esempio compare in entrambi i topic). Sebbene nell'ultimo topic compaiano parole come "cute" e "kid" che potrebbero consentire di classificare le recensioni della classe "Children", questo non sembra risultare sufficiente.

Prendendo atto che il metodo LSA non è stato in grado di discriminare i topic, si procede quindi ad analizzare i risultati del metodo LDA.

LDA

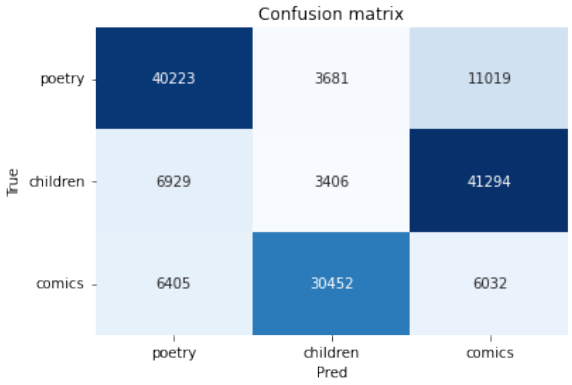
L'algoritmo sembra identificare i topic corretti relativamente ai 3 generi di libri: Poetry, Children e Comics. Topic 0 comprende tutti i topic relativi al genere "Children" sono presenti infatti parole come: child, young, little, kid, school, boy. Il topic 1 invece fa riferimento al genere "Poetry" infatti sono presenti parole quali: poem, poetry, poet, collection. Infine il Topic 2 fa riferimento al genere "Comics" infatti vediamo parole come: comic, series, graphic, art, love, think, fell.
Sicuramente sembra migliorare la capacità di identificare i topic corretti rispetto al metodo LSA.

Una volta ottenuti i topic mediante LDA, possiamo costruire un dataframe in cui viene evidenziato il topic associato ad ogni documento con maggiore probabilità.

	Topic0	Topic1	Topic2	dominant_topic
Doc0	0.900000	0.050000	0.050000	0
Doc1	0.930000	0.030000	0.030000	0
Doc2	0.600000	0.040000	0.360000	0
Doc3	0.780000	0.110000	0.110000	0
Doc4	0.800000	0.010000	0.190000	0
Doc5	0.840000	0.090000	0.070000	0
Doc6	0.580000	0.060000	0.360000	0
Doc7	0.600000	0.380000	0.020000	0
Doc8	0.970000	0.020000	0.020000	0
Doc9	0.620000	0.040000	0.350000	0
Doc10	0.980000	0.010000	0.010000	0
Doc11	0.100000	0.820000	0.080000	1
Doc12	0.180000	0.800000	0.020000	1

Confrontiamo ora i risultati ottenuti con i veri valori presenti nella colonna "genre".

	Topic0	Topic1	Topic2	dominant_topic	genre
Doc0	0.90	0.05	0.05	children	children
Doc1	0.93	0.03	0.03	children	children
Doc2	0.60	0.04	0.36	children	children
Doc3	0.78	0.11	0.11	children	children
Doc4	0.80	0.01	0.19	children	children
...	...	...	...	...	...
Doc149436	0.01	0.01	0.98	comics	poetry
Doc149437	0.01	0.20	0.79	comics	poetry



I risultati ottenuti dalla LDA non sono soddisfacenti, viene classificato correttamente solo il genere "Poetry", cattive performance si ottengono relativamente ai generi "Children" e "Comics". Inoltre si ottengono valori di coerenza e di perplessità molto alti, anche se queste misure sono difficilmente interpretabili.

Rispetto a LSA, tuttavia sembra che LDA riesca ad individuare meglio i topic. Questi consentono infatti di identificare in modo abbastanza chiaro i tre generi di riferimento dei documenti.

Capitolo 6

Conclusioni

In conclusione vediamo che per ciò che riguarda il task di Text Classification, la Logistic Regression sembra fornire buone performance in entrambe le rappresentazioni. Viceversa lo SGD che nel caso della rappresentazione TF-IDF sembrava uno dei migliori modelli, produce cattivi risultati utilizzando una rappresentazione Word2vec.

In termini generali si può dire che la rappresentazione TF-IDF sembra fornire risultati più soddisfacenti rispetto all'utilizzo della rappresentazione word2vec.

Per ciò che concerne il task di Topic Modeling, invece la tecnica LSA non sembra individuare correttamente i topic, non consentendo di discriminare tutte e tre le tipologie di documenti. Al contrario la tecnica LDA sembra fornire dei topic adeguati ai generi. Nonostante ciò la matrice di confusione non sembra fornire buoni risultati, infatti la tecnica LDA consente di classificare correttamente solo il genere "Poetry".

Sicuramente è possibile migliorare le performance in entrambi i task prendendo in considerazione rappresentazioni differenti. Un ulteriore passaggio che potrebbe migliorare le performance riguarda l'implementazione di ulteriori tecniche di pre processing che potrebbero consentire di mantenere le sole parole maggiormente significative, quindi per esempio potrebbe essere utile aggiungere alle stop words delle parole aggiuntive. Altri miglioramenti potrebbero derivare dall'utilizzo di dataset che siano maggiormente rappresentativi.