PROGETTO TEXT MINING AND SEARCH

Stefano Daraio
Corso di Laurea in Data Science
Università di Milano Biccoca
Matricola 718443
s.daraio@campus.unimib.it

Silvia Bordogna
Corso di Laurea in Data Science
Università di Milano Biccoca
Matricola 736610
s.bordogna2@campus.unimib.it

July 13, 2019

ABSTRACT

Negli ultimi anni sulla terra sono stati prodotti più dati rispetto a quanto fatto in tutta la storia precedente. Il volume di dati prodotti, però, non è di per sè sufficiente a garantire uno sviluppo della conoscienza, vi è anzi la necessità di sintetizzare questa enorme mole di bit per ricavarne informazione. Una delle sfide più interessanti e complesse riguarda il natural language processing per via del formato *semi-unstructured* o *unstructured* in cui si presenta. In questo progetto si intende applicare tecniche proprie di questo ambito ad un contesto reale quale quello dei contributi degli utenti su piattaforme web, nello specifico si tratteranno le recensioni di prodotti lasciate su Amazon. L'obiettivo è quello di risolvere il seguente problema: predirre la categoria merceologica di un prodotto a partire dalla sua recensione. Il progetto è diviso in due fasi, la prima fase di pre processing necessaria a rendere il linguaggio naturale un oggetto utilizzabile e comprensibile dalla macchina, e la seconda fase di implementazione e validazione del modello di classificazione.

1 Introduzione

Amazon ha rivoluzionato la modalità di acquisto del XXI secolo. Nata nel 1994 come e-commerce di soli libri, aveva come punto di forza il fatto di proporre una vastissima lista di titoli. Pochi anni dopo la fondazione, Jeff Bezos decide di permettere agli utenti di lasciare commenti sugli acquisti fatti ed è proprio questo aspetto, unitamente allo sviluppo di un *recommeder system* per proporre nuovi prodotti, che hanno creato una modalità di acquisto innovativa e di successo. Per quanto riguarda lo *user generated content*, trattandosi di campi *free text*, può essere utile dover implementare algoritmi che ne valutino la rilevanza con il prodotto a cui si riferiscono, per questo può essere interessante indovinare la categoria a cui appartiene un prodotto a partire dalla recensione.

1.1 Obiettivo

Si è deciso di svolgere il task di classificazione sviluppando un modello che predica la categoria merceologica di un prodotto a partire dalla sua recensione. Obiettivo finale è ottenere i migliori risultati sui dati di test.

2 Dati

I dati utilizzati sono direttamente forniti in formato json dalla Standford University al seguente sito:

`https://snap.stanford.edu/data/web-Amazon.html`

Si tratta di 142.8 milioni di recensioni in lingua inglese, datate tra Maggio 1996 e Luglio 2014. Per scopi simili a quelli da noi perseguiti, la piattaforma mette a disposizone e suggerisce l'utilizzo di un sample normalizzato, chiamato *5-core*, in cui si contano 5 recensioni per user e 5 recensioni per prodotto.

Vi sono 24 categorie di prodotto a disposizione. Ai fini del progetto, per motivi di limiti computazionli, sono state scelte le seguenti 10 categorie, cercando di utilizzare gruppi con il minor numero di osservazioni.

- Clothing, Shoes and Jewelry (278.677)
- Toys and Games (167.597)
- Tools and Home Improvement (134.476)
- Beauty (198.502)
- Office Products (53.258)
- Pet Supplies (157.836)
- Automotive (20.473)
- Digital Music (64.706)
- Musical Instruments (10.261)
- Amazon Instant Video (37.126)

I file di ciascuna categoria sono stati quindi uniti per un totale di 1.122.192 righe, aggiungendo a ciascuno una variabile contenente la categoria di appartenenza, che costituirà la variabile target del modello.

Inifne ciascun file json contiene i seguenti campi per ogni recensione *reviewerID, asin, reviewerName, helpful, review-Text, overall, summary, unixReviewTime, reviewTime* di cui noi useremo solo *review-Text* come variabile esplicativa del modello.

3 Metodologia

Il progetto consiste di due fasi, la prima parte ha l'obiettivo di trasformare il contenuto delle recensioni in un formato che possa poi essere processato e utilizzato come input nella seconda fase, durante la quale ci si è occupati di creare un classifier seguendo il metodo iterativo di allenamento su trainig e validation sul test.

3.1 Text pre-processing

In questa fase si sono effettuate delle trasformazioni su ciascun documento per ottenere una rappresentazione del testo utilizzabile dall'algoritmo di classificazione.

Nella prima fase di tokenization è stato necessario stabilire delle regole in base alla quali identificare sequenze di caratteri che costituiscano un unità sensata di testo.

I documenti sono inoltre stati normalizzati, ovvero il testo è stato trasformato in *lowercase* ed è stata rimossa la punteggiatura.

Successivamente si è passati alla fase di stemming, ovvero il processo che riduce ogni parola al suo lemma, di modo da ridurre la numerosità delle istanze mantendone però gran parte del significato. Si è scelto di optare per questo metodo, da cui si sono ottenuti già risultati soddisfacenti, piuttosto che usare il metodo lemmization per motivi computazionali.

Sempre tramite il pacchetto nltk si è scelto infine di rimuovere le *stop words*, ovvero i termini più comuni del linguaggio, per ridurre ulteriormente il numero di termini della matrice document-term.

3.2 Classification

Il dataset è stato quindi diviso in due partizioni corrispondenti al training set ed al test set. La partizione per il test set è stata pari al 10% delle osservazioni totali, ovvero 112.219 osservazioni. Tale percentuale risulta essere adeguata dato il numero elevato di osservazioni totali.

Abbiamo quindi proceduto con il processo di classification applicando la libreria XGBoost (eXteme Gradient Boosting). XGBoost è un algoritmo di machine learning basato su un albero decisionale che utilizza un framework di gradient boosting.

il processo di elaborazione dell'algoritmo comporta la creazione e l'aggiunta sequenziale di alberi decisionali, ognuno dei quali tenta di correggere gli errori dei learners che lo hanno preceduto.

Ciò solleva la questione del numero di alberi (weak learners o stimatori) da configurare nel modello di incremento del gradiente e di quanto dovrebbe essere grande ogni albero.

Rispetto al Gradient Boosting Machines (BGMs) ci sono miglioramenti in termini di :

1) Ottimizzazione di sistema ad esempio tramite l'implementazione parallela nella costruzione sequenziale di alberi o il criterio di potatura degli alberi.

2) Miglioramenti degli algoritmi tramite cross validation integrata ad ogni iterazione, gestione della sparsity awareness, metodi per prevenire l'overfitting ed altri elementi.

I parametri maggiormente significativi utilizzati nel modello sono i seguenti :

- max depth = 4 : Rappresenta la profondità di ogni albero, che è il numero massimo di diversi attributi utilizzati in ogni albero stesso, un aumento di tale valore ne aumenta la complessità ma anche il rischio di overfitting, in seguito a diverse prove abbiamo trovato il valore ottimale da attribuire.
- min child weight = 5 : Si tratta della somma minima dei pesi delle istanze necessarie nei child, ovvero rappresenta il parametro di arresto nella valutazione della fase di partizione dell'abero.
- objective = 'multi: softmax' : l'obiettivo di apprendimento viene definito tramite classificazione multiclasse usando il softmax objective.
- eval metric = mlogloss : come metrica di valutazione per i dati di validazione abbiamo scelto di usare, in quanto garante di risultati migliori, la multinomial log loss ,questa è la funzione di perdita definita come la probabilità log-negativa delle etichette reali date le previsioni di un classificatore probabilistico
- verbose eval = 10, in questo modo andiamo ad istruire l'agoritmo in modo che iteri il train finchè il mlogloss non migliori dopo 10 iterazioni

4 Risultati

L'accuracy complessiva del modello e' dell' 83%.

Si può notare come si siano riportati buoni risultati sui vari dataset di analisi, in particolare 7 classi su 10 riportano valori di precision e recall maggiori all'80%. Per quanto riguarda altre metriche di valutazione, i risultati ottenuti sono riassunti nella tabella sottostante:

	precision	recall	f1-score	support
0	0.94	0.91	0.93	3696
1	0.81	0.51	0.63	2053
2	0.91	0.90	0.91	19704
3	0.85	0.94	0.89	27926
4	0.38	0.42	0.40	6559
5	0.32	0.25	0.28	6520
6	0.92	0.86	0.89	5302
7	0.95	0.88	0.92	15761
8	0.80	0.84	0.82	13511
9	0.91	0.89	0.90	16704
micro avg	0.83	0.83	0.83	117736
macro avg	0.78	0.74	0.76	117736
weighted avg	0.83	0.83	0.83	117736

Si nota che le classi "music" ed "instruments" sono spesso confuse, questo probabilmente è dovuto alla somiglianza dei termini usati nelle due classi.

La classe "auto" invece non riporta buone performance, probabilmente quest è dovuto al fatto che si tratta di una classe sottorappresentata nel dataset. In particolare essa riporta una recall bassa, ovvero solo la meta' delle recensioni appartenenti alla categoria sono state correttamente assegnate, mentre l'altra meta' e' stata assegnata ad altre classi, in particolare "home". Questo potrebbe essere stato generato da un problema di class imbalance.

5 Conclusioni

Valutando le metriche ottenute dall'applicazione del modello possiamo concludere che riesce ad eseguire con un buon livello di precisione il task richiesto. Quindi ci aspettiamo che sia in grado di classificare con un buon margine di correttezza una qualsiasi review aggiuntiva non considerando le classi precedentemente individuate come problematiche. E' possibile migliorare ulteriormente la qualita' del modello in primis utilizzando un maggior numero di reviews per il training, ma questo implica una maggiore capacità e conseguente sforzo computazionale .Sarebbe anche possibile implementare un modello XGBoost (o comunque un classificatore) piu' articolato, ma nonostante alcuni test effettuati non abbiamo notato miglioramenti considerando la significativa potenza computazionale aggiuntiva richiesta.