Università degli studi di
Milano-Bicocca

Text Mining and Search

Amazon Reviews Classification

Authors:
Marco Ferrario - 795203 - m.ferrario39@campus.unimib.it
Giorgio Ottolina - 838017 - g.ottolina1@campus.unimib.it

Sommario

Lo scopo di questo progetto è elaborare algoritmi di machine learning basati sulla **classificazione multiclasse** di diverse **recensioni testuali di prodotti Amazon**, per catalogarle in base alla valutazione ricevuta. Si è utilizzato il linguaggio di programmazione Python.

1 Dataset

I dataset iniziali scelti per questo progetto riguardano tutti recensioni di prodotti Amazon di diverso tipo:

- **Prodotti elettronici**;
- **Kindle**;
- **CD musicali**;
- **Film/programmi televisivi**;

Il dataset complessivo ottenuto attraverso join dei suddetti gruppi di dati è costituito da **5466932 righe**, ed è in **formato JSON**. Le recensioni sono inoltre arricchite da metadati inerenti a features quali voto complessivo ("overall"), utilità ("helpfulness"), tempo, nome del recensore, ecc. Per affrontare il problema di classificazione verranno prese in considerazione le features **"Text"**, che contiene il testo della recensione e **"overall"**, la classe da predire (da 1, votazione minima, a 5, votazione massima).

2 Preprocessing

Prima di procedere col preprocessing del testo, si è pensato di scremare il Dataset a disposizione al fine di ottenere il subset piu significativo possibile. Per questo motivo, sono state selezionate le recensioni ritenute più utili in quanto caratterizzate da almeno 6 voti, più di 4 parole e in lingua inglese (identificate con la libreria **langdetect**). A seguire, sono state adottate le seguenti fasi di preprocessing:

- **Tokenization** - il testo viene suddiviso in singoli termini, o "token";

- **Normalization** - le parole del testo sono tutte ricondotte a una forma standard, e in particolar modo termini estremamente simili vengono uniformati. In questo passaggio viene considerato anche il lowercasing del testo per far diventare tutte le lettere minuscole;
- **Punteggiatura e Numeri** - vengono rimossi eventuali spazi e caratteri non necessari, insieme ai numeri;
- **Rimozione delle Stop Words** - sono rimosse le parole ritenute superflue e viene effettuato anche un conteggio delle Stop Words in questione;
- **Stemming** - questo passaggio è stato attuato implementando il **Porter Stemmer** importato dalla libreria di NLP per Python "nltk".
- **Lemmatization** - step ulteriore rispetto allo Stemming per cercare di risalire a radici il più attendibili possibile delle parole

Il Dataset, a questo punto, contiene 872304 recensioni, con una media di 145 parole per singolo testo. Attraverso delle visualizzazioni di tipo **wordcloud**, è stato possibile individuare i termini con frequenza di apparizione più elevata all'interno delle recensioni di ogni classe.

3 Text Representation

Per la fase di feature extraction and selection e quindi di rappresentazione del testo sono stati utilizzati i seguenti approcci:

- **Count Vectorizer** - impiegato per calcolare la frequenza delle parole in base a tutti i possibili unigram (parola singola) o bigram (coppie di parole);
- **Tf-Idf** - questa soluzione descrive il peso dei singoli termini non solo in base alla loro frequenza di apparizione all'interno di un documento ma anche in base alla loro importanza nell'intera collezione (corpus). Anche in questo caso sono stati considerati unigram e bigram;
- **Select KBest** - metodo di selezione degli attributi offerto dalla libreria SKlearn di Python che assegna uno score ad ogni feature, utilizzando una funzione. In questo caso è stato scelto di utilizzare il test del χ^2 con 10000 features (valore in grado di produrre i risultati migliori dopo alcuni tentativi).

4 Classification Models

Dopo aver eseguito la divisione del dataset ("Text" e "overall", quest'ultima è la feature su cui si basano le predizioni) nei subset di Train e Test (rispettivamente il 67% e il 33% del totale), sono stati implementati diversi modelli di machine learning per la classificazione. La classe presenta 130289 occorrenze per il valore 1, 83975 per il valore 2, 106065 per il valore 3, 160486 per il valore 4, 391489 per il valore 5. Si nota quindi un leggero sbilanciamento verso le recensioni valutate positivamente.

Gli algoritmi utilizzati sono stati:

- LinearSVC;
- Logistic Regression;
- Random Forest;
- Multinomial Naive Bayes;
- Gradient Boost.

Model	**Count Vectorizer**	**Tf-Idf**
SVC	Acc: 0.60 - Time: 24m	Acc: 0.61 - Time: 8m
Logistic Regression	Acc: 0.61 - Time: 30m	Acc: 0.61 - Time: 1m
Random Forest	Acc: 0.45 - Time: 39.5s	Acc: 0.45 - Time: 48.7s
Multinomial NB	Acc: 0.50 - Time: 745ms	Acc: 0.51 - Time: 622ms
Gradient Boosting	Acc: 0.50 - Time: 2m	Acc: 0.50 - Time: 2m

Infine, si è deciso di implementare una Cross Validation con 5 folds. Tutti i risultati sono discussi nella seguente e ultima sezione.

5 Evaluation and Conclusion

Gli algoritmi di classificazione che hanno offerto le prestazioni migliori sono stati **Logistic Regression** e **Linear SVC**. In termini di accuracy, il valore massimo raggiunto è stato il 60% con questi due modelli. Per quanto riguarda precision e recall si è osservata una certa difficoltà a classificare correttamente

le recensioni valutate con 2, 3 e 4 stelle. La difficoltà nella classificazione per quanto riguarda queste recensioni è probabilmente da imputarsi all'elevata frequenza di parole simili tra di loro all'interno delle suddette classi. Si può facilmente notare come i modelli con rappresentazione effettuata tramite If-Idf siano stati caratterizzati da tempi di esecuzione notevolmente più rapidi rispetto alle controparti di tipo Count Vectorizer, sebbene non abbiano avuto risultati molto più significativi in termini di effectiveness. Nel dettaglio, la riduzione più significativa si è potuta osservare negli algoritmi LinearSVC e Logistic Regression, dove il training è stato effettuato rispettivamente in 8 e 1.5 minuti, contro i 26 e 32 minuti con CountVectorizer. Nota a parte deve essere fatta in merito al modello Random Forest, che purtroppo non risulta utile dati i valori nulli delle altre metriche (in particolar modo della Recall). Infine, è stata applicata la **Cross Validation** con 5 folds: il tempo di esecuzione è logicamente incrementato dato il maggior numero di iterazioni da compiere (divisione tra train e test e vettorizzazione per ogni fold), e non sono stati riscontrati miglioramenti degni di nota nelle metriche di effectiveness dei modelli.

Il problema principale da affrontare nella realizzazione del task è stato il processare una grande quantità di dati (oltre 800 000 recensioni). Ciò ha limitato l'uso di alcuni algoritmi di classificazione, che in alcuni casi sono stati implementati in modo essenziale. Per possibili miglioramenti, si potrebbero utilizzare ulteriori metodi di Feature Extraction - Selection/Synthesis – Weighting, come ad esempio Word2vec, mutual dependence, Matrix decomposition.