Analisi del sentimento
Naive Bayes e Perceptron

Alessio Chen

1 Introduzione

In questo elaborato andiamo a testare due algoritmi di Machine Learning per l'analisi del sentimento, un campo del Natural Language Processing che si occupa di interpretare e classificare opinioni e sentimenti all'interno di un testo attraverso alcune tecniche di analisi testuale computazionale. In particolare, viene utilizzato il dataset "Large Movie Review Dataset" [1], il quale contiene recensioni di film.

2 Dataset

2.1 Caratteristiche

Il dataset utilizzato contiene 50.000 recensioni suddivise in modo uniforme: 25.000 per l'addestramento (train) e 25.000 per il test.
In tutta la collezione, non sono ammesse più di 30 recensioni per ogni dato film perché le recensioni per lo stesso film tendono ad avere un rating correlato.

2.2 Files

Ci sono due cartelle di primo livello [train/, test/] che contengono rispettivamente i file per il training e il testing. Ognuna di queste due cartelle contiene al suo interno due cartelle [pos/, neg/] per le recensioni, le quali sono memorizzate all'interno di file di testo denominati secondo la seguente convezione: **[[id]_[rating].txt]**. Dove [id] è un codice univoco che identifica un film e [rating] è la valutazione in stelle per il film espressa su una scala da 1 a 10. Per esempio, il file [test/post/200_8.txt] è il testo per una recensione positiva per il film con id 200 e valutazione 8/10.

2.3 Bag of Words (BoW)

Oltre ai file di testo contenente le recensioni, all'interno del dataset troviamo anche una bag of words già tokenizzata pronta per l'uso, memorizzata nei file .feat nelle cartelle train/test. Ogni file .feat è in formato LIBSVM, un formato vettoriale per l'etichettatura dei dati. Ognuno di questi file contiene 25.000 righe, dove ogni riga rappresenta il valore dell'etichetta e il vettore delle caratteristiche delle occorrenze delle parole per la corrispondente recensione nel training/testing set.
Per esempio la seguente riga è la prima riga contenuta in train/labeledBow.feat

9 0:9 1:1 2:4 .... 47304:1

Il primo 9 sta a indicare che la prima recensione ha dato un punteggio di 9, 0:9 indica che nella recensione il token n.0 (corrispondente alla parola "the" che troviamo all'interno del file imdb.vocab) ha 9 occorrenze all'interno della recensione, 1:1 indica che c'è una occorrenza della parola "and", 2:4 indica che ci sono 4 occorrenza della parola "a" e 47304:1 indica che c'è una unica occorrenza della parola "pettiness".
Il vettore di input descritto qui sopra calcola fondamentalmente il numero di occorrenze per ogni parola/token che appare nel testo di una recensione.

3 Algoritmi di classificazioni utilizzati

3.1 Naive Bayes

Il classificatore Naive Bayes utilizza il Teorema di Bayes, il quale fornisce un modo per calcolare la probabilità di un'ipotesi in base alla nostra precedente conoscenza. Questo algoritmo si base sull'assunzione "naive" che l'effetto dell'attributo su una data classe è indipendente condizionalmente dai valori degli atri attributi; quindi la presenza o l'assenza di una particolare feature non influenza la presenza o l'assenza di altre caratteristiche. Il problema della classificazione è quindi posto in termini probabilistici. Grazia a questa assunzione è possibile considerare:

$$P(a_1, a_2, ..., a_n|v_j) = \sum_{i=1} P(a_i|v_j)$$

In questo progetto si è utilizzato l'implementazione multinomiale disponibile nella libreria scikit-learn [2]

3.2 Perceptron

Il Perceptron è un classificatore di tipo binario, il quale dato un dataset contenente n features x prova a imparare una funzione lineare del tipo:

$$f(x) = \begin{cases} 1 & \text{se } \ wx + b > 0 \\ 0 & \text{altrimenti} \end{cases} \tag{1}$$

dove w è un vettore di pesi e quindi $w * x$ è il prodotto vettoriale

$$\sum_{i=0}^{n} w_i x_i$$

n è il numero di input e b è il fattore di Bayes. L'algoritmo itera su tutti gli esempi presenti nel train set finché la funzione non predice correttamente tutte le loro labels, altrimenti i parametri della funzione sono aggiornati nel seguente modo: $w = w + yx$ e $b = b + y$. Nell'altra eventualità si può passare all'input successivo poiché la classificazione risulta secondo la funzione. Se il dataset non è linearmente separabile questo algoritmo non è in grado di terminare e quindi di classificare correttamente.

4 Procedimento

L'elaborato si divide in due esperimenti, nel primo esperimento andiamo a utilizzare la bag of words già presente nel dataset per testare i due classificatori. Mentre nel secondo andiamo a creare una nostra bag of words utilizzando i dati presenti nel dataset.

4.1 Esperimento 1

Inizialmente è stata utilizzata la funzione *load_svmlight_files()* presente nella libreria di scikit-learn, la quale legge i dati di training e testing e li salva in una matrice. Per poter utilizzare gli algoritmi di classificazioni su questi dati è necessario eseguire una normalizzazione del vettore delle feature,la normalizzazione utilizza in questo caso è lo schema TF-IDF (term frequency–inverse document frequency).
Dopo aver letto i file, analizzato i dati e calcolato la metrica TF-IDF , il passo successivo è l'addrestramento di un modello di classificazione per differenziare i film con feedback positivi e negativi. Per fare ciò con il metodo *binireze()* andiamo a convertire le recensioni con punteggio $<= 5$ come negative, mentre quelle con punteggio > 5 come positive. Infine con il metodo *classifie()* andiamo ad addestrare un modello utilizzando il classificatore che riceve in input e calcolarne l'accuratezza. L'implementazione di tutti e due i modelli utilizzato sono presenti in scikit-learn.

4.2 Esperimento 2

In una prima fase si leggono i dati di training e testing contenuti nei file .txt utilizzando la libreria os di python. A questo segue una fase di preprocessing, che ci permette di "pulire" e semplificare il testo, vengono eliminati dai testi i numeri, i caratteri speciali, la punteggiatura e ogni carattere alfabetico viene trasformato in minuscolo. A questo punto il testo è suddiviso in parti più piccole (tokens) e si estrae la bag of words. Per fare ciò viene utilizzato la funzione *TfidfVectorizer()*, presente in scikit-learn, la quale restituisce una rappresentazione matriciale dell'intero testo, costruendo un dizionario di tutte le parole presenti nel testo per ogni documento in base alla metrica TF-IDF. Il testo viene quindi convertito in una matrice sparsa di tokens.
Per rappresentare ogni recensione si utilizza un approccio n-grams, ovvero oltre alle singole parole, vengono utilizzate anche le coppie di parole per rappresentare il testo. Questo è possibile grazie al parametro *ngram_range()* presente in *TfidfVectorizer()* che permette di definire il minimo e il massimo di parole da considerare per la rappresentazione del testo.
Inolte, per ridurre la dimensione del dizionario eliminiamo i termini che hanno una frequenza maggiore di 0.9, ovvero quei termini che appaiono in più del 90% dei documenti, grazie al parametro *max_df()* presente in *TfidfVectorizer()*.
Gazie al parametro *binary* di *TfidfVectorizer()* impostato a True, abbiamo una matrice sparsa delle features in forma binaria: 1 se è presente almeno un'occorrenza di quel token nel documento, 0 altrimenti.
Infine, vengono inizializzati i due diversi classificatori , utilizzando le implementazioni presenti entrambi in scikit-learn, addestrandoli sui dati del training-set precedentemente estratti e utilizziamo per l'analisi e la predizione il test-set.

5 Risultati

Di seguito sono riportati i risultati ottenuti nei due esperimenti:

	Naive Bayes Accuracy	Perceptron Accuracy
Esperimento 1	83.04	84.92
Esperimento 2	87.58	88.69

Tabella 1: Tabella comparativa dell'accuratezza dei due esperimenti

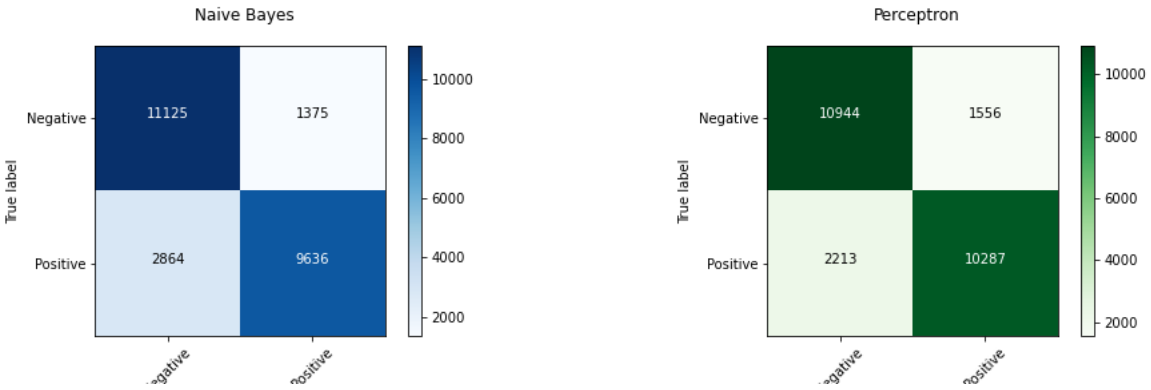

Figura 1: Matrici di confusione per Esperimento 1

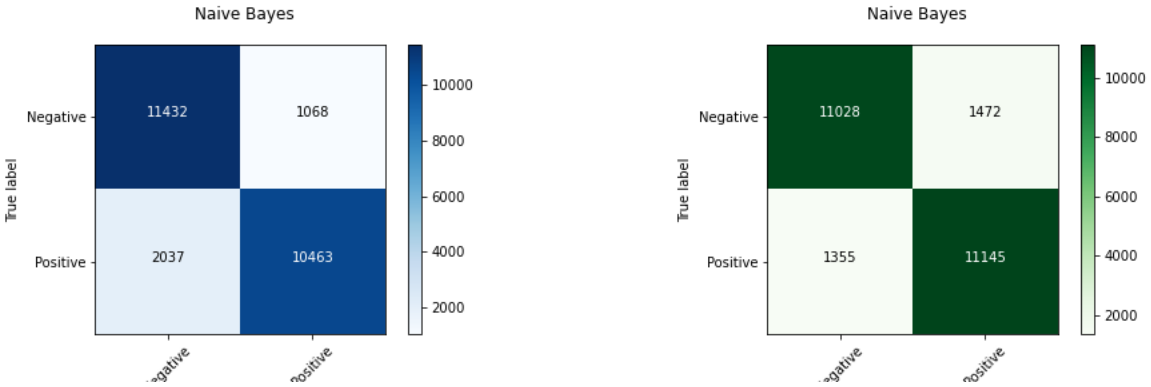

Figura 2: Matrici di confusione per Esperimento 2

Riferimenti bibliografici

[1] Large Movie Review Dataset v1.0: `http://ai.stanford.edu/~amaas/data/sentiment/`

[2] Scikit-Learn: `https://scikit-learn.org/stable/user_guide.html`

[3] Learning Word Vectors for Sentiment Analysis: `http://ai.stanford.edu/~amaas/papers/wvSent_acl2011.pdf`