Relazione Progetto

Giovanni Corradini[1] e Matteo Franzolin[2]

[1] Corso di laurea magistrale in Scienze Statistiche, matricola 1210593
giovanni.corradini.1@studenti.unipd.it
[2] Corso di laurea magistrale in Scienze Statistiche, matricola 1242775
matteo.franzolin@studenti.unipd.it

Sommario: in questo progetto sono stati implementati due diversi modelli statistici di classificazione binaria, il percettrone ed il support-vector machines (SVM), con lo scopo di classificare adeguatamente delle frasi di recensioni di libri, nelle classi "frase spoiler" e "frase no-spoiler". L'approccio seguito è di tipo empirico e quindi finalizzato all'implementazione dei due modelli con successivo esperimento effettuato su dati testuali. Entrambi i modelli denotano una discreta capacità di riconoscere le frasi spoiler di una recensione (il SVM ne "riconosce" 4 su 5), ma comunque sembrano essere altri gli approcci preferibili per risolvere questo tipo di problema.

Keywords: percettrone - SVM - classificazione di testi - spoiler

1 Introduzione

Il web è una miniera di informazioni. Tra queste si trovano le recensioni testuali, scritte da persone, riguardanti un prodotto od un servizio da loro utilizzato. A differenza della maggior parte dei beni e servizi che acquistiamo, della quale vorremmo conoscere ogni dettaglio e giudizio prima di procedere con la spesa, per i libri ed altri contenuti multimediali non è così: infatti, vi è il rischio che una recensione contenga uno spoiler, ossia un'anticipazione su un qualche avvenimento presente nel prodotto. A differenza dei contenuti no-spoiler, apprezzati perché forniscono una panoramica generale del prodotto alla quale siamo interessati, i contenuti spoiler spesso rovinano l'esperienza del consumatore.

Nei problemi relativi alla classificazione di testi spesso si ha a che fare con un numero di variabili esplicative molto grande. Infatti, per classificare dei testi spesso si utilizzano come variabili esplicative le occorrenze (oppure una misura derivata da esse) di una determinata parola all'interno di un testo.
Per rendere i modelli più stabili e robusti, per ridurre la complessità del problema e per strutturare i dati grezzi, è preferibile dunque filtrare, integrare, ridurre e trasformare i dati attraverso una fase cosiddetta di preprocessing.
All'interno della procedura di preprocessing si svolge anche la procedura di features selection: attraverso test statistici o altre metodologie specificatamente ben definite, viene selezionato solamente un sottoinsieme dell'insieme delle variabili esplicative di partenza, spesso mantenendo solamente quelle variabili che più discriminano tra testi appartenenti a classi diverse.

In questo progetto vengono implementati un percettrone e un SVM con kernel di tipo lineare, che verranno poi applicati ad un insieme di dati relativo alle

frasi testuali presenti nel dataset *Goodreads*. Questi due modelli statistici di apprendimento supervisionato sono efficaci in quanto separano correttamente dei dati linearmente separabili; la conferma è stata data dall'applicazione dei due modelli ad un dataset linearmente separabile.

Un classificatore binario basato sul percettrone converge solamente se i dati sono linearmente separabili, ovvero se esiste un iperpiano che separa tutti i punti di una classe da tutti i punti dell'altra. Se nessun iperpiano separatore esiste, il percettrone non riesce a convergere e si rende necessario fissare un numero massimo di iterazioni per arrestare l'esecuzione dell'algoritmo.
A differenza del percettrone, il SVM non ha questa limitazione, in quanto converge comunque ad un qualche separatore che, purché non perfetto se i dati non sono linearmente separabili, produce il miglior classificatore possibile rispetto ad una ben definita metrica. Infatti, l'algoritmo sottostante al SVM si basa sulla ricerca dell'iperpiano separatore ottimale, che è il classificatore lineare che crea il più largo margine possibile tra le due classi. Questa ricerca si riduce ad un problema di minimizzazione di una funzione di costo che può essere risolto tramite diverse tecniche di ottimizzazione convessa, sia esatte che approssimate. Nell'esperimento del progetto è stato scelto di utilizzare lo stochastic gradient descent (SGD), una tecnica di ottimizzazione che, rispetto al batch gradient descent, porta a risultati approssimati ma in tempi nettamente minori, soprattutto quando si ha a che fare con una grande mole di dati.

Tra i vantaggi che presenta il percettrone rispetto al SVM, vi è il minor tempo di esecuzione impiegato dall'algoritmo per la stima dei parametri e la possibilità di essere aggiornato quando si ha un nuovo dato a disposizione, senza ripetere da capo tutta la procedura di stima (algoritmo online). Inoltre, il percettrone implementato nell'esperimento al capitolo 5, è stato adattato a dei dati non linearmente separabili, dunque non si rende necessario implementare il parametro relativo al learning rate in quanto esso non altera i risultati predittivi, ma influisce solamente sulla velocità di convergenza dell'algoritmo.

2 Base di partenza

In letteratura è stato recentemente proposto da Wan et al. un modello denominato *SpoilerNet*[3], che si prefigge l'obiettivo di classificare delle frasi di recensioni di prodotti multimediali.
L'approccio seguito dagli autori si basa su una accurata analisi dei dataset, che ha portato ad interessanti risultati per quanto riguarda le variabili da tenere in considerazione per lo sviluppo del modello poi proposto, che si basa su una rete neurale ricorrente progettata ad hoc per il problema.
Uno dei problemi evidenziati dagli autori di *SpoilerNet*, che hanno applicato i loro modelli sul dataset *Goodreads*, riguarda la presenza di una controversa etichettatura di alcuni contenuti spoiler, in quanto nel dataset ogni frase è etichettata come spoiler o meno dall'autore della recensione. A causa del diverso campione utilizzato per lo scopo, i risultati ottenuti da *SpoilerNet* non sono direttamente confrontabili con i risultati ottenuti nell'esperimento mostrato al capitolo 5.

3 Problema affrontato

3.1 Dimensione e tipologia dei dati

Il dataset *Goodreads*[4] si presenta in formato .json.gz: il formato JSON (acronimo di JavaScript Object Notation) è un formato testuale molto comune e pratico da elaborare. Per ridurre la dimensione del file contenente il dataset, i fornitori di esso lo hanno compresso tramite tecnologia GZIP (dimensione file compresso di 592 MB). I dati presenti nel dataset sono per la maggior parte di tipo testuale.

Il dataset *Goodreads* è composto da 1378033 recensioni relative a 24558 diversi libri. In particolare, il dataset è composto da una lista di 1378033 dizionari.
Ogni dizionario contiene le seguenti chiavi:

user_ id → ha come valore una stringa rappresentante il codice univoco dell'utente che ha scritto la recensione;

timestamp → ha come valore una stringa rappresentante la data della pubblicazione della recensione (in formato aaaa-mm-gg);

review_ sentences → ha come valore una lista. Ogni lista è composta da due elementi: il primo di tipo intero che assume valore 0 se la frase non contiene spoiler e 1 altrimenti, il secondo di tipo stringa rappresentante una frase della recensione;

rating → ha come valore un intero da 1 a 5, rappresentante il voto che l'utente ha dato al libro nella sua recensione (1 voto minimo, 5 voto massimo);

has_ spoiler → ha come valore un booleano che assume valore 'False' se non ci sono frasi contenenti spoiler nella recensione, 'True' se è presente almeno una frase contenente spoiler;

book_ id → ha come valore una stringa contenente un numero intero che rappresenta il codice identificativo univoco del libro;

review_ id → ha come valore una stringa rappresentante il codice univoco della recensione.

3.2 Analisi esplorativa

Le 1378033 recensioni totali sono composte per il 6.5% da recensioni contenenti almeno uno spoiler (89627) e per il restante 93.5% da recensioni con nemmeno uno spoiler (1288406). In media una recensione contiene 12.82 frasi.
Analizzando solamente le 89627 recensioni che presentano al loro interno almeno una frase spoiler, si può vedere come in esse siano presenti in media 23.78 frasi, di cui in media 17.42 sono no-spoiler e 6.36 sono spoiler.
Assumendo come unità statistica la frase, il corpus di testi dell'intero dataset ha una dimensione di 17672655 frasi. Di questo corpus totale, 569724 sono frasi spoiler (il 3.22% del totale), mentre le restanti 17102931 sono frasi no-spoiler. In media una frase spoiler contiene 16.33 parole, mentre una frase no-spoiler ne contiene 15.02.
Le 89627 recensioni che contengono almeno uno spoiler sono formate da un totale di 2131128 frasi, di cui 569724 sono frasi spoiler e 1561504 sono frasi no-spoiler, quindi un rapporto di circa 1 a 3. Le frasi no-spoiler contengono in media 15.04 parole mentre le frasi spoiler contengono in media 16.33 parole.

Come suggerito dagli autori di *SpoilerNet*, viene analizzata la posizione delle frasi spoiler all'interno delle recensioni (contenenti almeno uno spoiler). Si nota come le frasi spoiler tendano a concentrarsi nella parte finale di una recensione.

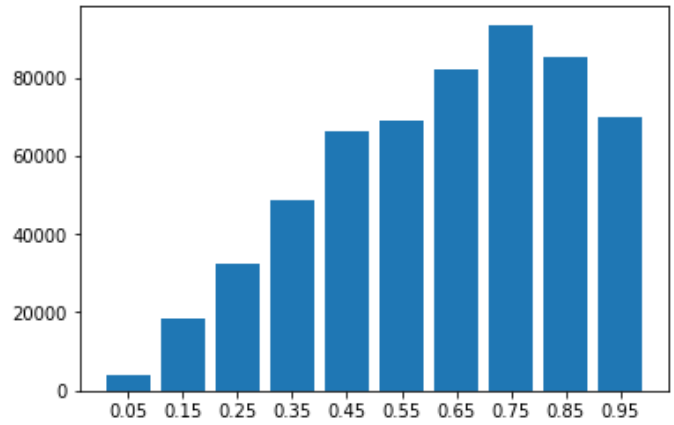

Figure 1: Posizione frasi spoiler all'interno della recensione

3.3 Efficienza degli algoritmi

Per confrontare l'efficienza dei due algoritmi di classificazione progettati (percettrone e SVM), sono stati adattati i due modelli a 10 diversi dataset di numerosità campionaria crescente, mantenendo fissato il numero di variabili esplicative, il numero di iterazioni degli algoritmi ed il valore degli iperparametri. Dal grafico sottostante si può notare come il tempo di esecuzione di entrambi gli algoritmi cresca linearmente all'aumentare della numerosità campionaria e come, a parità di fattori, il percettrone sia globalmente più efficiente del SVM.

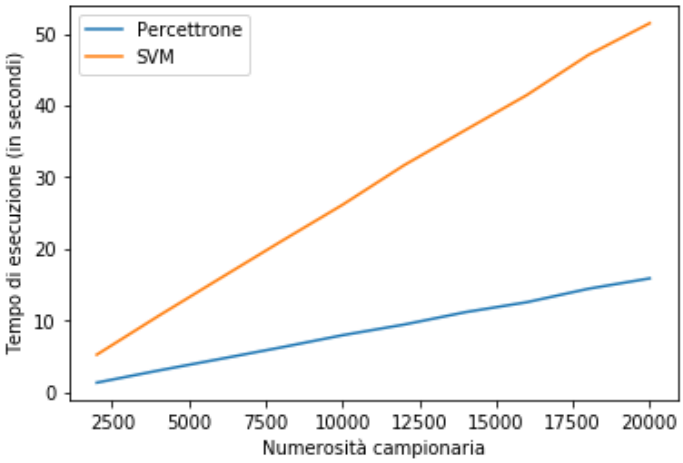

Figure 2: Confronto tra tempi di esecuzione

4 Metodi proposti

4.1 Test *chi quadrato* per features selection

Per determinare le variabili esplicative dei due modelli di classificazione implementati, è stato scelto di adottare un approccio *bag-of-words*. In altre parole, verranno utilizzate le occorrenze di alcuni termini all'interno della frase per classificarla come spoiler o no-spoiler.
Per diminuire la complessità computazionale del problema e per rendere i modelli più stabili e robusti, è stato deciso di implementare ed utilizzare il test *chi quadrato* per la selezione delle variabili da includere nei modelli. Questo test, che in generale è utilizzato per confrontare la distribuzione delle frequenze osservate con quella delle frequenze attese di una certa variabile, in questo contesto è stato applicato ad ogni termine preso singolarmente, confrontando le frequenze osservate e quelle attese di questo termine fra le due classi oggetto di studio (spoiler/no-spoiler). Più nel dettaglio, il test per ogni termine è il seguente:
siano oss_0 e oss_1 le frequenze del termine osservate rispettivamente nelle classi no-spoiler e spoiler, sia inoltre $att = \frac{oss_0 + oss_1}{2}$, la frequenza attesa di tale termine per entrambe le classi (in quanto è assunta l'ipotesi di equidistribuzione del termine fra le due classi).
Il punteggio del test per tale termine è ottenuto calcolando:

$$\frac{(oss_0 - att)^2}{att} + \frac{(oss_1 - att)^2}{att} \ .$$

A causa dello sbilanciamento fra le classi spoiler e no-spoiler, la somma delle frequenze assolute dei termini è diversa fra le due classi, dunque è stato deciso di utilizzare come frequenze nel test chi quadrato quelle relative, ovvero le frequenze assolute divise per la somma di tutti i termini, separatamente per le due classi.
Dopo aver applicato il test sulla distribuzione di ogni termine, si utilizza il punteggio ottenuto dal test (o il relativo p-value) per ordinare i termini; termini con un punteggio del test più elevato (p-value più basso) hanno una distribuzione fra le due classi meno uniforme, e quindi una capacità di discriminazione fra le due classi più elevata, rispetto a quelli con punteggio più basso.

4.2 Percettrone[5]

Il primo metodo di classificazione utilizzato per l'obiettivo del progetto è il percettrone, un modello di classificazione binaria che ricerca un iperpiano in grado di separare (linearmente) lo spazio delle variabili esplicative, di modo tale che le variabili riferite alla 'classe positiva' stiano dalla parte positiva dell'iperpiano e quelle riferita alla 'classe negativa' dalla parte negativa del iperpiano.
L'algoritmo di stima del percettrone, applicato su un insieme di dati di allenamento, è il seguente:

sia data una matrice $X_{n \times d}$ contenente i predittori, un vettore d-dimensionale y contenente la variabile risposta (classi +1 e -1), un vettore d-dimensionale w contenente i coefficienti e una soglia $\theta \in R$. Siano inoltre x_i l'i-esima riga della

matrice X e y_i l'i-esimo elemento del vettore y.

(1) Si inizializza $w = 0$, con w d-dimensionale
(2) Per $n\_iter = 1, 2, \ldots$:
per $i = 1, 2, \ldots, n$:

Se $w * x_i > \theta$ allora $y_i^* = +1$
Se $w * x_i \leqslant \theta$ allora y_i^* = -1
Se y_i^* è uguale a y_i allora non fare nulla
Se y_i^* è diverso da y_i allora $w = w + (y_i * x_i)$

Dato che la convergenza dell'algoritmo, ovvero il momento in cui tutte le unità statistiche sono correttamente classificate, non è assicurata se l'insieme di allenamento non è linearmente separabile, è stato deciso di non impostare un numero massimo di iterazioni né qualche criterio di arresto, ma di trattare il numero di iterazioni come un iperparametro da ottimizzare.
Inoltre, dato che a priori non esiste un valore ottimale della soglia θ, è stato scelto di incorporarlo nel vettore dei coefficienti w ($w = (w, \theta)$) e di aggiungere una covariata, costantemente uguale a -1, affinché la soglia diventi uno dei coefficienti che viene allenato col procedere delle iterazioni.
L'unica variazione da apportare all'algoritmo per tenere conto di questa modifica consiste nell'assegnare $y_i^* = +1$ se $w * x_i > 0$ e assegnare y_i^* = -1 se $w * x_i \leqslant 0$.

4.3 SVM[5]

Il secondo metodo di classificazione utilizzato per l'obiettivo del progetto è il support-vector machines, il quale, a differenza del percettrone, non solo ricerca uno dei possibili iperpiani che separi lo spazio delle esplicative, ma ricerca l'iperpiano separatore ottimale che crea il più largo margine possibile tra le due classi.
L'algoritmo di stima del SVM, implementato su un insieme di dati di allenamento e che come metodo di ottimizzazione sfrutta lo stochastic gradient descent, è il seguente:

sia data una matrice $X_{n \times d}$ contenente i predittori, un vettore d-dimensionale y contenente la variabile risposta (classi +1 e -1), un vettore d-dimensionale w di coefficienti, un parametro di regolazione $C \in R$ e un parametro di apprendimento $\eta \in R$. Siano inoltre:

- x_i l'i-esima riga della matrice X
- y_i l'i-esimo elemento di y

- $J(w) = \frac{1}{N} \sum_{i=1}^{N} [\frac{1}{2} ||w||^2 + C * max(0, 1 - y_i * (w * x_i))]$
la funzione di costo da minimizzare

- $\bigtriangledown_w (J(w))_i = \begin{cases} w \text{ se } max(0, 1 - y_i * (w * x_i)) = 0 \\ w - (C * y_i * x_i) \text{ altrimenti} \end{cases}$

il gradiente della funzione di costo da minimizzare, calcolato solamente con l'i-esima osservazione (si noti la differenza dal batch gradient descent nel quale il gradiente viene calcolato su tutte le osservazioni).

(1) Si assumono η e C fissati e $w = 0$, con w d-dimensionale
(2) Per $n\_iter = 1, 2, \ldots$:
    per $i = 1, 2, \ldots, n$:
        calcolare $\bigtriangledown_w(J(w))_i$
        $w = w - (\eta * \bigtriangledown_w(J(w))_i)$

Dato che la convergenza dell'algoritmo, ovvero il momento in cui tutte le unità statistiche sono correttamente classificate, non è assicurata se l'insieme di allenamento non è linearmente separabile, è stato deciso, analogamente a quanto fatto con il percettrone, di non impostare un numero massimo di iterazioni né qualche criterio di arresto, ma di trattare invece il numero di iterazioni come un iperparametro da ottimizzare.

5 Esperimenti

5.1 Preprocessing dei dati

Lo scopo del preprocessing dei dati è quello di riorganizzare i dati grezzi a disposizione, in maniera tale da ottenere la matrice X di variabili esplicative ed il vettore y di variabili risposta che verranno utilizzati come input per il training, la validation ed il test dei modelli.
In particolare, il dataset Goodreads è composto da una lista di dizionari, ognuno avente 7 chiavi con i relativi valori. In questa sottosezione si elencano le procedure effettuate per rielaborare i dati con lo scopo di ottenere una matrice X composta dal numero di occorrenze di ogni feature selezionata in ogni frase (osservazione), ed il vettore y di variabili risposta, composto da più zeri e uni.

Selezione dei dati:
Come scritto in precedenza, il 6.5% delle recensioni contiene almeno una frase spoiler, mentre il restante 93.5% non ne contiene nemmeno una. In relazione agli scopi del progetto, per ovviare al problema di sbilanciamento delle classi e per ridurre il costo computazionale dell'intera procedura, è stato deciso di utilizzare solamente un campione delle recensioni, ovvero quello formato da recensioni contenenti almeno una frase spoiler e relative a libri recensiti almeno 20 volte. Riassumendo:

- sono stati selezionati i dizionari relativi alle 89627 recensioni contenenti almeno una frase spoiler
- sono stati selezionati i dizionari relativi alle 24798 recensioni dei 574 libri recensiti almeno 20 volte

Integrazione di variabili:
Grazie all'intuizione avuta dagli autori di SpoilerNet, ossia che le frasi spoiler tendono a presentarsi alla fine di una recensione, fatto confermato dall'analisi esplorativa, è stato deciso di inserire una variabile relativa al posizionamento della frase all'interno della relativa recensione. Inoltre è stato deciso di inserire

delle variabili, sotto forma di fattori, che codificano a quale libro è associata ogni frase, in quanto si è notato che le recensioni tendono ad essere libro-specifiche (es. nomi di personaggi). Riassumendo:

- è stata aggiunta ad ogni frase una variabile che assume valori da 0 a 1, indicante la posizione della frase all'interno della recensione
- sono state aggiunte ad ogni frase 573 variabili che assumono valore 0 o 1, indicanti il libro alla quale si riferisce la recensione (un libro sarà codificato con un vettore di zeri per evitare problemi di multicollinearità).

Riduzione dei dati:
A questo punto, da ogni dizionario sono state estratte le frasi con le relative variabili, creando dunque una nuova lista di liste, in cui ogni sottolista contiene le informazioni relative ad una frase. In particolare, ogni sottolista ha come elementi: il valore dicotomico indicante se la frase è spoiler o no-spoiler, la stringa rappresentante la frase, il valore indicante la posizione della frase all'interno della recensione, i valori che codificano ogni libro come fattore.
Sono poi stati eliminati da ogni frase i caratteri speciali e le stop words, ossia parole che spesso vengono usate all'interno delle frasi, ma che sono poco significative (es. articoli, congiunzioni).
Per ridurre la complessità del problema e per ottenere modelli più stabili e robusti, è stato deciso di applicare lo stemming alle parole: questa procedura, che "taglia" una parola fino alla sua radice, consente di ridurre il numero di features senza una perdita significativa di informazioni.
Inoltre è stato deciso di eliminare le frasi con meno di 10 termini e quelle con più di 30 termini: le frasi troppo corte sono spesso sinonimo di spam o comunque generalmente sono poco informative, inoltre non considerarle riduce la sparsità della matrice con aumento della stabilità dei modelli, mentre le frasi troppo lunghe spesso non sono vere e proprie frasi ma intere recensioni. Riassumendo:

- sono state estratte le frasi con le relative variabili
- sono stati eliminati dalle frasi i caratteri speciali e le stop words
- è stato applicato lo stemming alle parole
- sono state eliminate le frasi con meno di 10 termini o più di 30 termini, passando così da 636861 frasi a 175505 frasi.

Trasformazione dei dati:
Nel dataset ridotto sono presenti 54752 frasi spoiler e 120753 frasi no-spoiler, dunque le due classi hanno un peso rispettivamente del 31.2% e del 68.8% sul totale. La lista completa è composta da 175505 sottoliste, ognuna la quale è composta da 576 elementi (variabile risposta e stringhe rappresentanti la frase, oltre alle variabili di posizione e fattoriali menzionate in precedenza). Prima di procedere oltre, è stata divisa questa lista in due liste di ugual dimensione, così da poter utilizzare un insieme di dati per selezionare le features sulla quale si baseranno i modelli (features_ selection_ set), ed un altro insieme di dati per il training, la validation ed il test dei modelli (model_ set). In questa maniera si cerca di contenere il fenomeno dell'overfitting dei modelli ai dati.
Per procedere con la selezione dei predittori, per prima cosa sono state salvate le stringhe rappresentanti le frasi dell'insieme "features_ selection_ set" in un file in formato txt. Le stringhe sono state dunque processate tramite MapReduce, con lo scopo di ottenere le occorrenze di ogni diverso termine all'interno dell'insieme

di frasi in un tempo ragionevole.
E' stato poi deciso di mantenere solamente 9825 termini tra i 33932 diversi termini totali, non includendo tra le features quelli che si presentano più raramente (meno di 5 volte). Questa scelta porta a modelli più affidabili in quanto i termini comparsi molto raramente in passato spesso non hanno un significato letterale ben preciso.
A questo punto, è stata creata una matrice avente come elementi i conteggi di ogni termine selezionato all'interno di ogni frase. E' stato poi applicato il test *chi quadrato* ai vettori contenenti le occorrenze dei 9825 diversi termini, confrontando le frequenze dei termini comuni tra le due classi (spoiler e no-spoiler). Lo scopo del test è quello di ottenere la lista dei termini che più discriminano tra le frasi spoiler e quelle no-spoiler. Si è deciso di mantenere solamente il 20% dei termini che ottengono un punteggio più alto nel test (1853 termini), oltre ai 559 termini che compaiono solamente in una classe e non nell'altra, selezionando così in totale 2412 termini. Alcuni tra i termini selezionati che più discriminano tra le due classi sono: *review, reccomend, die, kill, end, dead.*
I 2412 termini ottenuti verranno definitivamente utilizzati come features per il model_ set. Viene quindi definita la "matrice" X: essa è composta dalle occorrenze di ogni termine (selezionato tramite *test chi quadrato* a partire dal features_ selection_ set) all'interno di una frase del model_ set, dalla variabile di posizione e dalle 573 variabili fattoriali relative al libro.
La "matrice" (lista di liste) X è dunque formata da 87753 sottoliste (numero di frasi del model_ set), ognuna composta da 2986 elementi (features). Questa "matrice", insieme al relativo vettore delle variabili risposta y formato da 87753 elementi, verranno utilizzati per la procedura di modellazione. Si nota che, a causa della brevità delle frasi e dell'elevato numero di features, la "matrice" delle esplicative è molto sparsa, ovvero presenta un ingente numero di zeri. Riassumendo:

- è stata suddivisa la lista in features_ selection_ set e model_ set
- tramite MapReduce sono state conteggiati i termini all'interno delle frasi del features_ selection_ set
- sono stati selezionati i termini che si presentano almeno 5 volte
- è stata creata una matrice avente come elementi i conteggi di ogni termine selezionato all'interno di ogni frase
- è stato applicato il test *chi quadrato* e sono stati selezionati i termini più significativi
- è stata creata la "matrice" delle variabili esplicative X ed il vettore delle variabili risposta y, utilizzando i dati del model_ set

5.2 Applicazioni: percettrone e SVM

Prima di procedere con la fase di modellazione, sono stati suddivisi la "matrice" X ed il vettore y in 3 insiemi secondo la proporzione 50/25/25 (numerosità 43846, 21938 e 21939), così da poter utilizzare insiemi di dati diversi per il training, la validation ed il test dei modelli.
Essendo l'obiettivo dell'esperimento quello di riuscire ad intercettare il maggior numero di frasi spoiler, piuttosto che riuscire a discriminare al meglio le frasi fra le due classi spoiler e no-spoiler, come misura prevalente delle performances dei modelli si è scelto di utilizzare la recall, ossia, relativamente a questo esper-

imento, il rapporto tra frasi spoiler individuate dal modello sul totale delle frasi spoiler.
Come ulteriore misura delle capacità predittive del modello è stato utilizzato l'F1-score, ossia la media armonica tra precisione e recall, calcolata quindi come:

$$F1\text{-}score = \frac{2 * precisione * recall}{precisione + recall} \ .$$

Per adattare ai dati dei modelli di classificazione progettati, essi necessitano dell'assegnazione di determinati valori ai propri iperparametri.
Per il percettrone vi è da determinare il numero di iterazioni, mentre per il SVM, oltre al numero di iterazioni, vi sono anche da determinare il learning rate (indicato con η nella descrizione dell'algoritmo) e l'iperparametro che regola l'ampiezza dei margini di classificazione del modello (indicato con C).
A causa della impossibilità di separare linearmente i dati a disposizione, aumentare il numero di iterazioni non porta ad avere dei risultati stabili, dunque per entrambi i modelli si è scelto di trattare il numero di iterazioni come un iperparametro da ottimizzare.
Per determinare gli iperparametri ottimali, sono stati stimati più modelli sul training set assegnando ogni volta dei differenti valori ai vari iperparametri; sono stati dunque selezionati quei valori che massimizzano il valore della recall sul validation set (composto da dati non utilizzati in nessun'altra procedura).
Successivamente, utilizzando i valori ottimali degli iperparametri, sono state misurate le capacità predittive di entrambi i modelli sul test set.

5.2.1 Percettrone

Per scegliere il valore ottimale del numero di iterazioni del percettrone, è stato stimato più volte il modello sul training set, variando il numero di iterazioni da 5 a 50, a step di 1. Il grafico seguente riporta la recall e l'F1-score ottenuti dal percettrone sul validation set, al variare del numero di iterazioni:

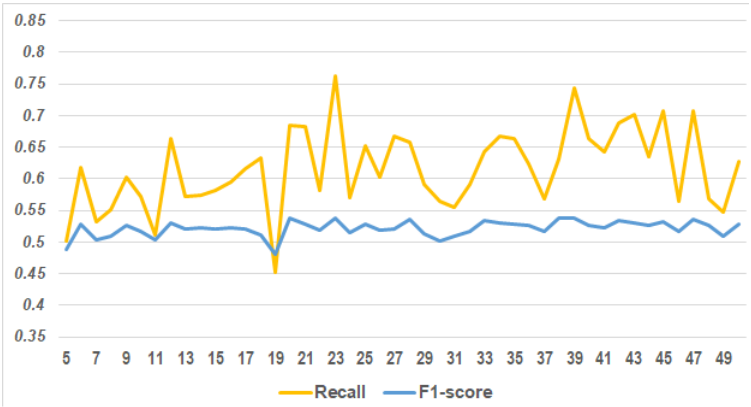

Figure 3: Confronto performances percettrone al variare del numero di iterazioni

Visti i risultati ottenuti, si è deciso di fissare a 23 il numero di iterazioni, valore che massimizza la recall sul validation set. Sono state dunque stimate le risposte sul test set, ottenendo una recall pari a 76% ed un F1-score pari a 54%, risutati

che indicano una buona capacità di riconoscere gli spoiler di questo modello, unita però ad una sua non altrettanto buona capacità di discriminare tra spoiler e no-spoiler, come si nota dalla matrice di confusione seguente:

	previsti	
osservati	**0**	**1**
0	7903	7137
1	1649	5250

Figure 4: Matrice di confusione percettrone

5.2.2 SVM

Per selezionare i 2 iperparametri e il numero di iterazioni ottimali per il SVM, si è deciso in primo luogo di mantenere fisso il numero di iterazioni a 23, valore ottimale scelto per il percettrone, misurando la recall e l'F1-score del modello per tutte le combinazioni di η e C, con il primo che varia tra $1E$-09 e $1E$-04 e il secondo tra $1E$+05 a $1E$+12, entrambi a step di un ordine di grandezza. Il grafico seguente riporta le perfomances ottenute dal SVM sul validation set, al variare di η e C, con numero di iterazioni fissato a 23:

	η					
C	*1E-09*	*1E-08*	*1E-07*	*1E-06*	*1E-05*	*1E-04*
1E+05	0.0500	0.3778	0.4309	0.5477	0.4719	0.4626
1E+06	0.3783	0.4330	0.5491	0.5825	0.5981	0.6443
1E+07	0.4355	0.6226	0.6348	0.6704	0.6705	0.5386
1E+08	0.5892	0.5529	0.6902	0.6201	0.4665	0.6132
1E+09	0.5305	0.7294	0.6692	0.5482	0.5223	0.4891
1E+10	0.6466	0.6055	0.5886	0.6818	0.6444	0.4891
1E+11	0.5363	0.7123	0.6836	0.5747	0.6444	0.4891
1E+12	0.6378	0.5031	0.6454	0.5747	0.6444	0.4891

Figure 5: Heatmap valori di recall, numero di iterazioni fissato a 23

La coppia di iperparametri $\eta = 1E$-08 e $C = 1E$+09, è quella che massimizza il valore della recall sul validation set, pari a 73%.
A questo punto, è stato deciso di mantenere il valore di η fissato e di stimare nuovamente il modello sul training set, facendo variare C in un suo intorno e variando il numero di iterazioni da 5 a 50, a step di 1; i valori che massimizzano la recall sul validation set (80%) risultano essere: $\eta = 1E$-08, $C = 1.1E$+09 e numero di iterazioni pari a 34. Con gli iperparametri ottimali selezionati sono state dunque stimate le risposte sul test set, ottenendo una recall pari a 80% ed un F1-score pari a 53%, risultati che indicano una capacità predittiva del SVM superiore a quella del percettrone (per quanto concerne l'intercettazione delle frasi spoiler), ma che presenta la stessa difficoltà riguardo alla discriminazione complessiva tra le due classi.

	previsti	
osservati	**0**	**1**
0	6762	8278
1	1410	5489

Figure 6: Matrice di confusione SVM

5.2.3 Conclusioni e spunti futuri

Riguardo all'efficienza delle procedure implementate, esse non hanno un costo computazionale eccessivamente elevato, ma l'utilizzo di una procedura basata sull'architettura MapReduce ridurrebbe notevolmente il tempo di esecuzione dei due algoritmi, soprattutto per quanto concerne la fase di selezione degli iperparametri. Riguardo l'efficacia degli algoritmi, l'obiettivo di ottenere un buon valore di recall è stato raggiunto da entrambi i modelli, tuttavia, in un contesto come l'analisi statistica testuale a cui è associata una complessità molto elevata (frasi brevi e numero di features molto elevato con varianza molto piccola), per raggiungere risultati migliori in termini di capacità predittive dei modelli, probabilmente è preferibile non utilizzare un approccio *bag-of-words*, ma piuttosto un approccio basato sul *word embedding*, integrandolo a modelli più complessi come, per esempio, le reti neurali ricorrenti.

5.3 Riferimenti bibliografici

[3] "Fine-Grained Spoiler Detection from Large-Scale Review Corpora" (Wan M., Misra R., Nakashole N., McAuley J.) - *https://arxiv.org/pdf/1905.13416.pdf*

[4] Dataset: "goodreads_reviews_spoiler"
Homepage: *https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/reviews*
Download link: *https://drive.google.com/uc?id=196W2kDoZXRPjzbTjM6uvTidn6aTpsFnS*

[5] "Mining of Massive Dataset" (Leskovec J., Rajaraman A., Ullman J.D.)

Altri testi consultati:

"Machine Learning in Automated Text Categorization" (Sebastiani F.) - *https://arxiv.org/pdf/cs/0110053v1.pdf*

"Search Engines - Information Retrieval in Practice" (Croft W.B., Metzler D., Strohman T.) - *https://ciir.cs.umass.edu/downloads/SEIRiP.pdf*