Rilevazione di messaggi spam con algoritmo Naive-Bayes

Luca Zanetti
matricola nr. 808229
luca.zanetti2@studenti.unimi.it

Sommario

L'individuazione di messaggi spam costituisce uno dei più noti esempi di classificazione di documenti testuali. Poiché i documenti possono essere suddivisi per i nostri scopi in due sole classi, i documenti *spam* e i documenti *non spam*, il problema può essere visto come un problema di classificazione binario.
In questa relazione verrà prima discusso in generale l'algoritmo utilizzato, Naive-Bayes, un algoritmo piuttosto semplice ma molto efficace per il problema oggetto di studio. Dopodiché verrà discussa l'implementazione realizzata ed infine i risultati ottenuti sperimentando la nostra implementazione su un particolare dataset.

1 L'algoritmo Naive-Bayes

L'algoritmo Naive-Bayes è un algoritmo piuttosto semplice ma che si è appurato essere in grado di comportarsi molto bene per problemi di classificazione di documenti testuali. In particolar modo, da studi pubblicati, si è rivelato molto efficace per la rilevazione di messaggi spam.
Nave-Bayes si occupa di classificare un documento in una delle classi possibili assumendo che ogni documento sia stato generato da un modello probabilistico, i cui parametri variano per ogni classe di documenti. Naive-Bayes cerca quindi di stimare la probabilità che quel documento sia stato generato da ciascuna classe, applicando la regola di Bayes per effettuare la classificazione, ovvero scegliendo la classe che con maggior probabilità ha generato tale documento.

Più formalmente si assume che il documento d da classificare sia generato da un *mixture model* avente come vettore di parametri θ. Il modello ipotizza che il documento venga generato innanzitutto scegliendo una delle m classi c_i con probabilità *a priori* $P(c_i|\theta)$ e, scelta la classe, il documento sia generato con probabilità $P(d|c_i, \theta)$. Di conseguenza la probabilità di generare il documento d è pari a

$$P(d|\theta) = \sum_{i=1}^{m} P(c_i|\theta) * P(d|c_i, \theta)$$

L'algoritmo, dato in input il documento d, restituisce quindi l'indice della classe c^* che più probabilmente è stata scelta per generare il documento conoscendo appunto il documento che è stato generato, ovvero:

$$c^* = \underset{i \in 1,\ldots,m}{\operatorname{argmax}} P(c_i|d)$$

dove $P(c_i|d)$ è calcolata in questo modo:

$$P(c_i|d) = \frac{P(c_i|\theta) * P(d|c_i, \theta)}{P(d|\theta)} \propto P(c_i|\theta) * P(d|c_i, \theta)$$

I parametri del modello vengono stimati a partire da esempi correttamente classificati, il cosiddetto *training set*, costituendo quindi un algoritmo di *apprendimento supervisionato*. Il modello probabilistico utilizzato in questa implementazione è il modello *multinomiale* che verrà descritto nel prossimo paragrafo.

2 Il modello multinomiale

Al fine di stimare la probabilità che una classe abbia generato un dato documento è necessario scegliere il modello probabilistico che ha generato tale documento. Uno tra i più popolari ed efficaci modelli per la classificazione del testo è il cosiddetto modello multinomiale. Esso prende il nome dalla distribuzione multinomiale con la quale ogni parola del documento secondo questo modello viene generata. La distribuzione multinomiale non è altro che la generalizzazione a più variabili della distribuzione binomiale nella quale però ogni prova indipendente non ha solo due possibili esiti, ma più esiti possibili ciascuno con diversa probabilità di verificarsi.
Il modello multinomiale rappresenta un documento come un vettore di

interi d che contiene il numero di occorrenze nel documento di ciascuna parola del vocabolario V con cui il documento è stato generato. Il vettore d, dopo aver scelto la classe con cui generarlo, è quindi estratto da una distribuzione multinomiale (caratteristica per la classe scelta) con tante prove indipendenti quante sono le parole di d. Ogni prova costituisce l'estrazione di una parola dal vocabolario. La probabilità di estrarre una determinata parola in ogni prova è un parametro, da stimare, che varia da classe a classe.
Si noti che sono state fatte due assunzioni piuttosto forti, *naive* appunto, da cui il nome dell'algoritmo: la scelta della lunghezza del documento si suppone indipendente dalla classe che lo ha generato ed ogni parola estratta non dipende dalle parole estratte precedentemente; di conseguenza la probabilità che in un documento compaia la parola x non è influenzata dal fatto che si conosca il numero di occorrenze della parola y nello stesso documento, supponendo però di conoscere la classe c che lo ha generato. I due eventi di conseguenza non sono necessariamente indipendenti, ma condizionatamente indipendenti.
Queste assunzioni sono naturalmente non realistiche poiché appare ovvio che le occorrenze di una parola in un documento possano essere influenzate dalle occorrenze di un'altra. Tuttavia esse permettono di semplificare notevolmente il modello assunto e la stima dei suoi parametri, permettendoci di non considerare le correlazioni tra occorrenze differenti. Inoltre è stato rilevato sperimentalmente che tali assunzioni non pregiudicano nella maggioranza dei casi l'accuratezza del classificatore.

Sia V il vocabolario di parole, d il documento generato, N_w il numero di occorrenze in d della parola $w \in V$, allora la probabilità che il documento d venga generato dalla classe c corrisponde a:

$$P(d|c,\theta) = P(|d|) * |d|! * \prod_{w \in V} \frac{P(w|c,\theta)^{N_w}}{|N_w|!}$$

dove $|d|$ indica la lunghezza in parole del documento, $P(|d|)$ la probabilità che venga generato un documento lungo $|d|$ parole e $P(w|c,\theta)$ la probabilità che venga estratta la parola w se il documento appartiene alla classe c.

Poiché la probabilità che il documento sia lungo $|d|$ parole è indipendente per assunzione dalla classe che lo ha generato, per effettuare

la predizione possiamo limitarci a stimare quindi

$$\prod_{w \in V} \frac{P(w|c,\theta)^{N_w}}{|N_w|!} \propto P(d|c,\theta)$$

3 Stima dei parametri

Per effettuare la predizione è necessario stimare la probabilità $P(w|c_i,\theta)$ che la parola w venga estratta dalla classe c_i e la probabilità a priori $P(c_i|\theta)$ che venga scelta la classe c_i per generare un documento.
Nel nostro caso le classi considerate sono solamente due, la classe dei documenti spam e la classe di quelli non spam. Indicheremo con esempi positivi i documenti del training set classificati come spam, esempi negativi i restanti. Il vocabolario V è costituito da tutte le parole presenti nel training set oppure da un sottoinsieme delle stesse ottenuto tramite tecniche di *feature selection* che verranno descritte successivamente.

I parametri possono essere stimati con il metodo della massima verosimiglianza e si riducono quindi ad eseguire dei conteggi. Il metodo della massima verosimiglianza ci restituisce quindi questi stimatori:

$$P(c_i,\theta) = \frac{\text{nr. documenti etichettati con } c_i}{\text{nr. documenti totali}}$$

$$P(w|c_i,\theta) = \frac{\text{nr. occ. di } w \text{ in documenti di classe } c_i}{\text{nr. totale di occ. in documenti di classe } c_i}$$

Tuttavia questi stimatori non sono utilizzabili praticamente: una parola non osservata in esempi positivi avrà difatti come stima della probabilità di apparire in documenti classificati come spam 0; ma questa deduzione è un po' troppo forte: il fatto che non sia apparsa negli esempi positivi non implica che un documento classificato come spam non appartenente al training set non la possa contenere. Si incorrerebbe quindi in un caso di overfitting. Per questo motivo viene solitamente applicato il cosiddetto *smoothing laplaciano* che consiste nell'aumentare di una unità ciascun conteggio di occorrenze.
Applicando queste considerazioni la stima della probabilità di un'occorrenza della parola w in un documento di classe c_i è calcolata nel seguente modo:

$$P(w|c_i,\theta) = \frac{\text{nr. occ. di } w \text{ in documenti di classe } c_i + 1}{\text{nr. totale di occ in documenti di classe } c_i + |V|}$$

dove $|V|$ è la dimensione del vocabolario.

4 Feature selection

I dati che il classificatore implementato utilizza per formulare una predizione sono costituiti dalle occorrenze delle parole presenti nel training set. Tuttavia non è sempre la soluzione ottimale utilizzare tutte le parole che occorrono almeno una volta nel training set: è chiaro che le occorrenze di alcune parole, come l'articolo *the*, avranno meno potere discriminativo di altre, come ad esempio la parola *money*. Di conseguenza ridurre la dimensione del vocabolario utilizzato potrebbe consentire una migliore capacità di generalizzazione del nostro classificatore.

Il problema è quindi quello di trovare una misura del potere discriminativo delle varie voci del vocabolario ricavato dal training set. A tale scopo si è scelto di utilizzare, come descritto in [1], come misura del potere discriminativo di una parola t la *mutua informazione* tra una variabile casuale $C \in \{0, 1\}$ che indica da quale classe una determinata occorrenza della parola t è stata estratta e una variabile casuale W_t su tutte le occorrenze di t.
La mutua informazione $I(C, W_t)$ rappresenta la differenza tra l'entropia di C e l'entropia di C condizionata su W_t. Essa può quindi essere calcolata nel seguente modo:

$$I(C, W_t) = H(C) - H(C|W_t) = \sum_{c \in \{0,1\}} \sum_{f_t \in \{0,1\}} P(c, f_t) \log\left(\frac{P(c, f_t)}{P(c) * P(f_t)}\right)$$

Dove $P(c)$ indica la probabilità che un documento (non) sia di classe c, $P(f_t)$ la probabilità che un'occorrenza di parola (non) sia costituita dalla parola t. Per stimare tali valori si è usato il metodo della massima verosimiglianza ottenendo degli stimatori che non sono altro che rapporti tra conteggi, analoghi agli stimatori presentati nel paragrafo 3.

5 Risultati sperimentali

L'algoritmo Naive-Bayes descritto è stato implementato in python e testato sul dataset *Ling-Spam* che raccoglie messaggi spam e non da

The LINGUIST List [1]. Il dataset è costituito da 2893 messaggi di cui 481 sono spam. Per il training sono stati utilizzati 2024 messaggi di cui 336 spam, mentre il testing è stato effettuato su 869 messaggi di cui 145 spam. L'algoritmo utilizza solamente il testo dei messaggi per eseguire la sua predizione, tralasciando il soggetto. Il numero totale di parole differenti che occorrono nel training test è pari a 44743.

Per misurare l'efficacia del classificatore sono stati presi in considerazione diversi indici: la *precisione*, ovvero il rapporto tra il numero di esempi positivi correttamente classificati e il numero totale di documenti classificati come positivi, il *recall*, ovvero il rapporto tra il numero di esempi positivi correttamente classificati e il numero totale di esempi positivi e la *F1 score*, la media armonica tra i due indici.
Indicando con tp e tn il numero di esempi, rispettivamente positivi e negativi, classificati correttamente e con fp e fn il numero di esempi classificati erroneamente come positivi e negativi e con P, R, $F1$ la precisione, il recall e la loro media armonica, possiamo calcolare i tre indici nel seguente modo:

$$P = \frac{tp}{tp + fp} \quad R = \frac{tp}{tp + fn} \quad F1 = 2 * \frac{P * R}{P + R}$$

I risultati ottenuti per differenti dimensioni del vocabolario sono mostrati nella tabella 1. Si può notare come il numero di falsi positivi e negativi sia decisamente basso per qualunque dimensione del vocabolario a conferma di come l'algoritmo seppur semplice sia adeguato a questo problema. Si può notare inoltre che non ci sono significative differenze se si varia la dimensione del vocabolario da 2000 voci in poi; ciò significa che le voci che possiedono il maggior contenuto discriminativo sono poche migliaia.
Allo stesso tempo però la tabella evidenzia come diminuendo la dimensione del vocabolario si ottengano risultati, seppur di poco, peggiori. Naturalmente queste considerazioni possono riferirsi solo al dataset utilizzato e di conseguenza non si esclude che l'operazione di feature selection possa migliorare le performance del classificatore su altre tipologie di dati (come mostrato ad esempio in [1]).

[1] http://linguistlist.org/

Dim. vocabolario	tp	tn	fp	fn	P	R	F1
1000	129	710	14	16	0.90	0.89	0.90
2000	132	714	10	13	0.93	0.91	0.92
5000	134	713	11	11	0.92	0.92	0.92
10000	135	715	9	10	0.94	0.93	0.93
20000	135	716	8	10	0.94	0.93	0.94
44743	135	718	6	10	0.96	0.93	0.94

Tabella 1: Performance del classificatore per dimensioni differenti del vocabolario.

6 Miglioramento della precisione

Nel precedente paragrafo sono stati analizzati i risultati ottenuti senza dare maggior peso ad errori di primo o secondo tipo (che abbiamo chiamato rispettivamente falsi positivi e falsi negativi). Tuttavia appare evidente che un buon classificatore deve sì cercare di individuare il più alto numero possibile di messaggi spam (limitando i falsi negativi), ma allo stesso tempo è necessario evitare per quanto possibile di classificare un messaggio come spam quando non lo è. Difatti errori di quest'ultimo tipo sono normalmente più dannosi di eventuali messaggi spam non individuati.

Per questo motivo è stato implementato un semplice schema che permette di abbassare il numero di falsi positivi, migliorando la precisione, a costo di aumentare il recall. Tale schema non fa altro che classificare un documento come spam solo nel caso che la stima ottenuta per la probabilità che il documento sia stato generato come spam superi di una certa percentuale la stima della probabilità che il documento sia stato generato come non spam. Tale percentuale è settata dall'utente. Chiamiamo c il valore decimale di tale percentuale.

In tabella 2 sono mostrati i risultati ottenuti per valori differenti di c e dimensione massima del vocabolario. Come si può notare valori crescenti di c determinano un aumento della precisione a scapito di un peggioramento, anche considerevole del recall.

I dati della tabella 2 sono mostrati poi in forma grafica in figura 1. Il grafico evidenzia come un valore di c di 0.01 riesca ad aumentare la precisione senza determinare un peggioramento sensibile del recall e della F1. Un valore di c di 0.03 permette di raggiungere un'elevata precisione, limitando i falsi positivi a 1, con una diminuzione del recall e della F1 comunque non drastica. Valori maggiori di c invece

Valore di c	tp	tn	fp	fn	P	R	F1
0	135	718	6	10	0.96	0.93	0.94
0.005	134	720	4	11	0.97	0.92	0.95
0.01	134	721	3	11	0.98	0.92	0.95
0.02	128	721	3	17	0.98	0.88	0.93
0.03	123	723	1	22	0.99	0.85	0.91
0.04	117	723	1	28	0.99	0.81	0.89
0.05	113	723	1	32	0.99	0.78	0.87
0.06	100	723	1	45	0.99	0.69	0.81

Tabella 2: Performance del classificatore al variare del parametro c.

determinano una ripida diminuzione sia del recall che, di conseguenza, della F1.

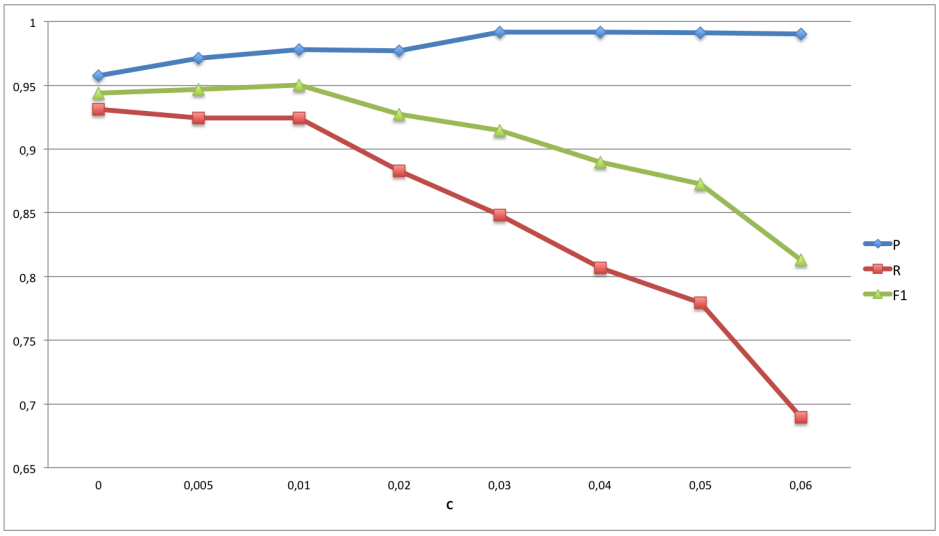

Figura 1: Precisione, Recall e F1-score al variare del parametro c

7 Utilizzo del classificatore

Per utilizzare il classificatore è necessario avere installato l'interprete python per la versione 2 del linguaggio. Se ne consiglia la versione 2.7.

Il programma è costituito dal file `nb.py` contenuto nella cartella `code`. Nelle cartelle `data/training` e `data/test` sono contenuti rispettivamente i messaggi utilizzati per l'addestramento e quelli per il testing. Ogni messaggio è costituito da un file di testo in cui la prima riga costituisce il soggetto, la seconda è vuota, mentre il corpo del messaggio inizia alla terza riga. In entrambe le cartelle i messaggi spam sono contenuti in file il cui nome inizia con un le lettere `spm`, mentre il nome dei file che contengono messaggi non spam inizia con un numero. Questa convenzione è stata ripresa dal dataset utilizzato.

Il classificatore può ricevere diverse opzioni da linea di comando. Se nessuna opzione è specificata il classificatore non effettuerà feature selection e adotterà un valore del parametro c pari a 0. Altrimenti per modificare il comportamento standard è necessario utilizzare le opzioni secondo il seguente schema:

./nb.py *valore di c* 2 *dimensione vocabolario*

Per utilizzare quindi un valore di c pari a 0.01 e una dimensione del vocabolario pari a 10000 è sufficiente invocare il programma nel seguente modo:

```
./nb.py 0.01 2 10000
```

Il programma stamperà quindi le sue predizioni e i risultati di conseguenza ottenuti.

8 Conclusioni

L'algoritmo Naive-Bayes è un algoritmo molto semplice; i risultati sperimentali hanno però confermato la sua efficacia nel problema di distinguere tra messaggi spam e non.
Si è osservato che le performance migliori con il dataset utilizzato si ottengono utilizzando come vocabolario tutte le parole che occorrono nel training set.
Infine si è ideato un semplice schema che permette di limitare il numero di messaggi non spam classificati erroneamente, a costo di un minor numero di spam rilevati; tale schema ha ottenuto risultati positivi, riuscendo a limitare considerevolmente il nuemro di falsi positivi senza compromettere la capacità del classificatore di rilevare messaggi spam.

Riferimenti bibliografici

[1] A. Mc Callum, K. Nigam (1998), *A Comparison of Event Models for Naive Bayes Text Classification*, Learning for Text Categorization: Papers from the 1998 AAAI Workshop, `http://www-2.cs.cmu.edu/~dgovinda/pdf/multinomial-aaaiws98.pdf`

[2] A. Ng, *Generative Algorithms*, Lecture notes of Stanford's *Machine Learning* course (cs229), `http://cs229.stanford.edu/notes/cs229-notes2.pdf`