Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in
Statistica e Tecnologie Informatiche

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

RELAZIONE FINALE

Sentiment Analysis con il metodo Hopkings-King: studio dei commenti ad un post di un blog

Relatore: Dott. Livio Finos
Dipartimento di Scienze Statistiche

Laureando: Andrea Domenico Girotto
Matricola N 583876

Anno Accademico 2012/2013

Indice

Introduzione

Vivendo in un'era in cui la tecnologia è alla portata di tutti, le informazioni relative ad eventi, cose e persone sono divenute chiare e veloci da reperire per tutti, soprattutto per i ricercatori, i quali non sono più costretti a richiedere informazioni persona per persona, ma possono trovare ciò che cercano direttamente in rete. Un grosso contributo in tal senso è dato dai social network come facebook e twitter.

I social network sono sempre più presenti nelle comunicazioni tra privati e, giorno dopo giorno, influenzano in maniera crescente la nostra esistenza. Le aziende piccole o grandi che siano, oggi giorno non possono più far a meno di pianificare le proprie strategie di comunicazione senza pensare ad integrare i social network come strumenti atti a costruire un dialogo con la propria clientela.

In Italia c'è ancora qualche azienda che non prende in considerazione l'utilizzo di questi nuovi strumenti di comunicazioni non vedendo vantaggi concreti o ritenendoli fuori dalla portata economica nonostante ci siano molte ricerche che lo smentiscano.[Anved e Aldim, 2013]

Ci troviamo davanti una grande contraddizione: da un lato abbiamo un insieme vastissimo di argomenti di discussione, di opinioni, di sentimenti e da qui un interesse crescente per monitorare tale varietà di informazioni. Dall'altro ci scontriamo con l'inadeguatezza delle tecniche di raccolta dati classiche, utilizzabili per dare conto di tale varietà di voci.

I sondaggi tradizionali (in qualunque forma: telefonica, on-line, ecc..) non si presentano come strumenti ottimali per far fronte al costante divenire della (cyber)opinione; sono costosi da progettare e implementare, sono per definizione statici (colgono nelle migliori ipotesi, lo stato di opinione su un dato argomento in uno specifico lasso di tempo). Se, ad esempio, tra il momento della progettazione del sondaggio e la sua implementazione emerge all'improvviso qualche fatto nuovo, il suo peso era stato trascurato nel sondaggio originario, quindi l'unico modo per darne conto è rifare l'intera operazione.

In più, i soggetti intervistati rispondono in modo “strategico”, mentendo o

travisando la propria reale opinione, in base a criteri che l'intervistatore non è in grado di controllare; quindi utilizzare un sondaggio tradizionale al giorno d'oggi per catturare l'opinione del web su un certo tema è dunque molto difficile.

Per fare questo, le aziende oggi hanno la possibilità di individuare, filtrare, analizzare e organizzare tutto quello che riguarda un settore soffermandosi sugli aspetti quantitativi e qualitativi che determinano l'aspetto positivo e negativo e il grado di emotività espresso grazie allo sviluppo di un metodo che è la Sentiment Analysis.

Con il termine Sentiment Analysis si riferisce a classificare le opinioni contenute in un testo scritto o parlato, tramite processi informatici, al fine di estrarre informazioni soggettive, opinioni e sentimenti dalle fonti di analisi osservate. La Sentiment Analysis è dunque lo strumento più usato per determinare il legame e i sentimenti (positivo, negativo, neutro e non attinente) del messaggio, tramite soprattutto facebook e twitter.

L'approccio tradizionale più usato per la Sentiment Analysis è tramite twitter, che è sicuramente il social network del momento. Sta vivendo in tutto il mondo una rapida espansione (solo in Italia 3,64 milioni di utenti attivi a Settembre 2012, +111% sull'anno precedente – fonte Audiweb/Nielsen), e con i suoi 140 caratteri porta l'utente ad andare dritto al punto senza giri di parole.

Twitter sembra dunque in grado di esprimere le reazioni della rete di fronte a qualsiasi evento (traumatico o lieto che sia), e commentando, reagendo in tempo reale le dichiarazioni rilasciate ad esempio da leader politici durante un dibattito televisivovo segnalandodo disservizi pubblici o festeggiando per la vittoria della proprio squadra del cuore, grazie alle nuove tecnologie (smartphone, tablet) .

Per l'analisi del sentimento di Twitter sono stati adottati dei metodi completamente automatizzati. Tra questi rientrano gli studi che contano ad esempio il numero di followers (cioè sono utenti di Twitter che decidono di seguire il vostro account su questo social network) di un profilo, di re-tweet comunemente abbreviato come RT, un retweet è un tweet che è stato ripetuto. Questo meccanismo viene spesso utilizzato per condividere notizie con i propri followers, oppure quelli che utilizzano un dizionario ontologico. In questo ultimo caso, si fa affidamento ad un vocabolario di parole definite ex-ante dal ricercatore

che sono associate da un lato al “polo negativo” e dall’altro al “polo positivo”. Sulla base della frequenza con cui un dato tweet utilizza tali parole, si definisce poi la polarità dello stesso (ad esempio pro o contro Beppe Grillo o Pierluigi Bersani, per rimanere all’ambito politico).

Entrambi gli approcci hanno evidenti limiti. Nel primo caso, non si distingue di che cosa parlano gli utenti Twitter su un dato tema: un elevato volume di citazioni, ad esempio, acquista un valore completamente differente a seconda se si parli (prevalentemente) bene o (prevalentemente) male del tema in questione. Nel secondo caso, il problema è che un dizionario ontologico per definizione non permette di catturare le sfumature presenti nel linguaggio, i doppi sensi, e l’evoluzione dello stesso nel tempo. Ad esempio, una frase come “La nuova linea della metro: ma che bella fregatura!” contiene al suo interno sia un termine positivo (“bella”) che uno negativo (“fregatura”). Un approccio automatizzato si troverebbe quindi in difficoltà ad identificare correttamente la posizione soggiacente di chi ha scritto questa particolare frase in relazione alla “nuova linea della metro”.

Un modo per uscire da questa fase complicata è adottare un approccio di Sentiment Analysis supervisionato. La tecnica di Sentiment Analysis che adottiamo è il metodo di analisi e monitoraggio proposto da Gary King e Hopkins Daniel J. Questa metodologia si basa su un processo a due fasi: nella prima fase un gruppo di codificatori (umani) analizza un sottocampione dei commenti pubblicati. Il sottocampione codificato servirà a identificare le categorie di opinione che verranno utilizzate per classificare i commenti di un post. La codifica umana permette di cogliere le sfumature all’interno dei testi, andando oltre la semplice individuazione dei termini positivi/negativi ed evitando di interpretare in maniera errata i post ironici. Il secondo *step* prevede che, attraverso un algoritmo statistico, la codifica venga estesa a tutti i post rimanenti che non erano stati letti.

Applicando questa metodologia ai post di Twitter, i ricercatori hanno condotto alcune analisi sui *trend* delle intenzioni di voto in occasione di diversi appuntamenti elettorali, ottenendo più volte risultati molto simili a quelli dei

sondaggi tradizionali , questo metodo verrà spiegato meglio nel primo e nel capitolo.

Nel terzo capitolo anziché usare messaggi di twitter noi analizzeremo i commenti rispetto ad un post di un blogs di Grillo. Abbiamo raccolto 6597 commenti ed a questi applicheremo il metodo proposto da Gary King e Holpkings e Daniel J., e poi confronteremo i risultati ottenuti tramite il metodo di alberi di classificazione [Chiogna, Pauli].

Capitolo I

1.1 Notazioni e Variabili

Per analizzare statisticamente il testo (cioè l'insieme dei nostri commenti del blog) bisogna rappresentare il linguaggio naturale come variabili numeriche che seguono procedure standard le variabili vengono calcolate seguendo 3 step, ciascuno dei quali funziona senza input umano ed è progettato per ridurre la complessità del testo.

I tre step sono i seguenti:

1) vengono eliminati “non English lunguage blogs” perchè considerate spam, cioè parole vuote, prive di significato ma servono solo per la costruzione della frase;
2) viene eseguita una pre-elaborazione del testo, all'interno di ogni documento tutte le parole vengono convertite in lettere minuscole, tolta la punteggiatura e ogni parola viene ricondotta alla sua radice originale. Esempio: le parole cartone, cartello, cartina, cartuccia diventano carta. Questa procedura chiamata “stemming” risulta più facile in lingua inglese mentre è molto più complicata nella lingua italiana;
3) il testo pre-elaborato viene riassunto in variabili dicotomiche, ogni variabile è caratterizzata dalla presenza / assenza di una determinata parola. Questa parola può essere costituita da:

    1° tipo: presenza / assenza di una parte denominata unigramma,
    2° tipo: presenza / assenza di due parti denominate bigramma,
    3° tipo: presenza / assenza di tre parti denominate trigramma,
    4° tipo: presenza / assenza di n parti.

La procedura richiede 2 insiemi di documento di testo:

- training-set: denominato insieme etichettato dove ogni documento i (i = 1,....,n) è etichettato con una data categoria D_i, aventi j modalità, tra i J valori possibili (J=1...j). Questo insieme verrà codificato a mano, ed

ogni singolo documento potrà assumere un unico valore D_i;

- test-set: è l' insieme di popolazione di tutti i documenti di testo che vogliamo classificare, dove ogni documento l (l=1,....L) verrà classificato utilizzando l'insieme del training-set in un secondo momento, per testare la validità del modello stimato.

Le altre variabili vengono definite dagli stessi documenti di testo del primo insieme denominate Word-stem e servono per definire la presenza o assenza di parola. Ogni Word-stem k è descritta dalla variabile S_{ik} dove i indica il documento appartenente e k la parola (word-stem) usata nell'iesimo documento.

$$S_{ik}=\begin{cases}1\ se\ la\ radice\ K(k=1,....,k)\ è\ usata\ almeno\ una\ volta\ nel\ documento\ i(i=1,....,n)\\ 0\ altrimenti\end{cases}$$

Per analizzare la presenza o l'assenza di queste word-stem si analizza il word stem profile Si, un vettore di lunghezza k che contiene tutte le parole utilizzate nel documento di testo, $S_i=(S_{i1},.....,S_{ik})$.
Quello che interessa all'analisi del documento è la percentuale totale di tutti i documenti che rientrano in ogni categoria: P(D)={P(D=1),.....P(D=J)}, dove P(D) è un vettore J*1, dove ogni elemento è calcolato da una tabulazione diretta:

$$P(D=j)=\sum_{l=1}^{L}1(D_l=i)$$

dove 1 è pari:

$$1=\begin{cases}1\ se\ è\ verà\\ 0\ altrimenti\end{cases}$$

D_i categoria di documenti è una variabile con diversi valori possibili che sono scelti dall'utente (-1 (negativo) , 0 (neutro), 1(positivo).....), Si costituisce un insieme di variabili dicotomiche. Ciò significa che P(D) è una distribuzione multinomiale con J possibili valori, e P(S) è una distribuzione multinomiale con 2^k valori possibili, ciascun valore risulta essere un word-stem profile.

Esistono due metodi per stimare P(D):

- il campionamento diretto: si individua una popolazione ben definita di interesse, si estrae un campione casuale della popolazione, si codificano a mano tutti i documenti del campione, e per finire si contano i documenti in ogni campione. Lo svantaggio principale è la perdita di tempo,inoltre il metodo fallisce quando il campione etichettato non è un campione casuale semplice.
- Aggregazione delle classificazioni dei singoli documenti (è di serie nella letteratura dell'apprendimento supervisionato): si utilizza un campione di documenti etichettato per stimare la funzione che lega la categoria, in cui andrà a classificarsi il documento D, e le word-stem S presenti nel documento. Questo metodo come il metodo descritto precedentemente fallisce quando il campione etichettato non è un campione casuale semplice, cioè quando l'insieme etichettato viene raccolto in un determinato periodo di tempo e lo si vuole applicare alla popolazione raccolta nel tempo.

  Inoltre il processo di generazione dati assunto dall'apprendimento supervisionato prevede D dato S, P(D|S), cioè prevedere le word-stem S data la categoria D, ma nel la realtà funziona al contrario cioè P(S| D). La conseguenza di utilizzare P(D|S) è il requisito di utilizzare due assunzioni necessarie per generalizzare dal campione etichettato alla popolazione.

  La prima assunzione è che S attraversa lo spazio di tutti i predittori D, il che significa che una volta che le variabili vengono controllate , non esiste nessun altra variabile che potrebbe migliorare il potere

predittivo. Nei problemi che coinvolgono il linguaggio umano, questa ipotesi non è soddisfatta poiché S è volutamente un'astrazione e quindi per definizione, non rappresenta tutte le informazioni esistenti nei predittori.

La seconda assunzione è che la classe di modelli scelti per P(D|S) comprende il vero modello, ma trovare il miglior o un buon modello è un compito difficile.

Capitolo II

2.1 Metodo ReadMe

L'approccio che utilizzo nella mia tesi per l'analisi dei sentimenti, è il metodo creato da Daniel J. Hopkings e Gary King, denominato ReadMe [Hopkings-Gary (2010)]. Questo metodo si basa sull' utilizzo della variabile S come conseguenza di D, cioè S è conosciuta e tramite questa informazione è possibile determinare D. In termini di probabilità abbiamo:

$$P(S=s)=\sum_{j=1}^{J} P(S=s|D=j)P(D=j)$$

Riassumendo in forma matriciale:

$$\underset{(2^k*1)}{P(S=s)}=\underset{(2^k*J)}{(S|D)}\underset{(J*1)}{P(D)}$$

con:

- P(S): indica la probabilità di ognuna delle 2^K possibili word-stem profile.
  Per esempio se K=4 word-stem, P(S) contiene le probabilità di ognuna della 2^3= 8 combinazioni possibili, da quella che non contiene nessuna word-stem a quella che le contiene tutte, quindi le combinazioni possibili sono, (000, 001, 010, 011, 100, 101, 110, 111)
- P(D): quantità / vettore d'interesse.
- P(S|D) è la probabilità di ciascuno degli word-stem profile che si verificano all'interno del documento con categoria j.

E' necessario attuare alcune assunzioni per proseguire l'analisi con questo metodo. L'assunzione più importante è considerare che P(S|D) dell'insieme etichettato, sia

uguale al P(S|D) della popolazione, così da poter stimare la matrice:

$$P^h(S|D) = P(S|D)$$

Attraverso questa osservazione, si stima P(D) conoscendo il valore P(S), tramite una regressione lineare. Viene considerato P(S|D) come variabile esplicativa X, P(S) come variabile indipendente Y e P(D) parametro da stimare β sconosciuto.
Quindi risulta la formula Y = Xβ, dove la probabilità che si verifichi ogni word-stem profile, è data dalla quantità P(D) e P(S|D).
Per stimare i coefficienti di regressione, quindi la variabile d'interesse si utilizza l'usuale formula:

$$\beta = (X^T X)^{-1} X^T y$$

Questo calcolo non serve per classificare i singoli documenti in categorie, per poi aggregarli per una sintesi generale, ma fornisce direttamente le proporzioni aggregate, la percentuale stessa della categoria.
La stima del vettore dei coefficienti Beta ci pone davanti a due problemi:

- K è molto grande, quindi un valore come 2^K sarà un numero enorme, impossibile da gestire anche per un computer.
- N << 2K , cioè il numero di osservazione per stimare P(S) e P(S|D) è molto più piccolo rispetto al numero delle word-stem profile.

Per risolvere questi problemi si sceglie casualmente un sottoinsieme tra le 5 e le 25 parole, attraverso il metodo di cross-validation. Una volta determinato il numero ottimale di parole per sottoinsieme, si risolve per ogni P(D) e è possibile fare la media dei risultati di ogni sottoinsieme.
Questo metodo è comunque adatto, poiché comporta assunzioni meno restrittive, ed è necessario solamente la distribuzione di P(S) e P(D).

2.2 Esempio

Tramite un esempio voglio spiegare al meglio il metodo descritto fin ora [Angela Andreella].

Da un dataset di tweet inerenti l'argomento 'Monti' selezioniamo 6 twitter come esempio:

1 ”Ecco il simbolo di #SceltaCivica #conMontiperlItalia alla Camera. ST #AgendaMonti http:t.coM8dbVnKW”

2 ”Caro # Monti riavvicinare i cittadini alla politica con Casini e Fini xe8 come attirare gli ermellini con l'aiuto di due pellicciai #ottoemezzo”

3 ”Un attimo ldots100.007 follower. WOW !! Benvenuti a voi e a quelli che verranno. #MontiLive @scelta_civica”

4 ”scelta_civica: Scarica il logo di #SceltaCivica su http:t.coKswArCdT! Rinnoviamo la politica, cambiamo l'Italia! #puoicontarci”

5 “Io sto con SenatoreMonti! Statista vero. Luciditxe0, intelligenza, competenza, serietxe0, visione e sobrietxe0 per @scelta_civica! @GLIOUTSIDER”

6 “SenatoreMonti @scelta_civica in italia la maggior parte degli evasori fiscali sono gli extra comunitari ke lavorano e investono il loro”

Questi corrispondo al testo originale, ora viene eseguita una pre-elaborazione del testo, cioè i messaggi vengono puliti.

Questa pulizia, realizzata da algoritmi consiste in varie fasi che comprende la trasformazione di tutte le lettere maiuscole in minuscole, eliminazione di articoli, congiunzioni e preposizioni.

1 “ecco simbolo #sceltacivica #conmontiperlitalia camera st #agendamonti wwwurlwww”

2 “caro #monti riavvicinare cittadini politica casini fini xe3xa8 come attirare ermellini aiuto due pellicciai #ottoemezzo”

3 “attimo 100 007 follower emotewow benvenuti voi quelli verranno

#montilive@ scelta_civica”

4 “rt @scelta_civica scarica logo #sceltacivica su wwwurlwww rinnoviamo politica cambiamo italia #puoicontarci”

5 “io sto senatoremonti statista vero luciditxe3 intelligenza competenza serietxe3 visione sobrietxe3 @scelta_civica@ glioutsider”

6 “rt senatoremonti @scelta_civica italia maggior parte evasori fiscali sono extra comunitari lavorano investono loro”

Si considerano 4 categorie di analisi:

-1 negativo 0 neutro 1 positivo 4 non attinente

Poiché ci sono 4 categorie, si ha J=4, i parametri di interesse da stimare risultano dunque in totale 4:

P(D=1) P(D=0) P(D=-1) P(D=4)

Nell'esempio si ha la seguente classifica

TWEET	CATEGORIA
1	1
2	-1
3	4
4	1
5	1
6	0

Tab. 2.1

Dopo aver categoralizzato ogni singolo tweet, si procede con l'analisi degli stem profile, infine si stima P(D). Dunque, si vede l'assenza o presenza delle determinate parole in ogni documento; non vengono prese tutte le parole di ogni documento (tweet) ma un suo sottoinsieme tra 5 e 10 parole. Esempio:
si prendono 5 per tweet. Per spiegare più facilmente l'esempio, si prende per ogni

documento una parola significativa, si hanno così 6 word stem profile, costituiti da 6 word stem, per un totale di 2^6 combinazioni= 32 combinazioni.
Dunque le rispettive 32 combinazioni: Si hanno dunque 32 possibili combinazioni di S, avendo P(S) e P(S|D) tale che

$$P(S=j)=\sum_{j=1}^{4} P(S=s|D=j)$$

In questo caso con j=4 categorie e k=6:

P(S = j) = (S = s|D = 1)P(D = 1)+P(S = s|D = -1)P(D = -1)+
+P(S = s|D = 0)P(D = 0)+P(S = s|D = 4)P(D = 4)

Il metodo ReadMe restituisce le percentuali di appartenenza ad ogni singola categoria, che in questo esempio sono -1, 0, 1 e 4.
Tabella con tutte le parole del sottoinsieme:

S6	S5	S4	S3	S2	S1	
0	0	1	0	0	1	#sceltacivica
0	0	0	0	0	1	#conmontiperl'italia
0	0	0	0	0	1	#agendamonti
0	0	0	0	0	1	simbolo
0	0	0	0	0	1	cmera
0	0	0	0	1	0	#monti
0	0	0	0	1	0	casini
0	0	0	0	1	0	fini
0	0	0	0	1	0	#ottoemezzo
0	0	0	0	1	0	pellicciai
0	0	0	1	0	0	followers
0	0	0	1	0	0	benvenuti
0	0	0	1	0	0	#montilive
0	0	0	1	0	0	emotewow

tab. 2.2

tabella con in evidenza gli word-stem profile e le categorie associate:

TWETT	CATEGORIA
S1	1
S2	-1
S3	4
S4	1
S5	1
S6	0

Tab 2.3

	#conmontiperl'italia	Casini	Followers	#puoicontarci	Competenza	evasori
S1	1	0	0	0	0	0
S2	0	1	0	0	0	0
S3	0	0	1	0	0	0
S4	0	0	0	1	0	0
S5	0	0	0	0	1	0
S6	0	0	0	0	0	1

Tab 2.4

Tabella con 32 possibili word-stem profile:

S	#conmontiperl'italia	Casini	Followers	#puoicontarci	Competenza	evasori
000000	0	0	0	0	0	0
...						
111111	1	1	1	1	1	1

Tab 2.5

CAPITOLO III

3.1 Applicazione ai dati reali

Le nostre analisi sono state fatte sullo studio di 6597 commenti, pubblicati sul blog di Grillo [beppegrillo]: “I figli di nn“ il cui testo è:

“Le nuove generazioni sono senza padri, sono figlie di NN, dal latino "*Nomen nescio : nome non conosco*". Sulle loro carte di identità, sui loro documenti di lavoro, nei libretti universitari alla voce "*figlio di*" risulta la sigla NN, figlio di nessuno, figlio della colpa, figlio di padre ignoto, figlio di vecchi puttanieri che si sono giocati ogni possibile lascito testamentario indebitando gli eredi. Non ci sono però responsabili conclamati della miseria, della mancanza di un futuro, di una qualunque prospettiva a cui sono stati condannati questi ragazzi. Nessuno ammette responsabilità di sorta. E' opera del destino cinico e baro, dello Spirito Santo, della moderna divinità chiamata mercato che si manifesta all'improvviso come un nume iroso che chiede sacrifici umani. Lo sfascio ha origini soprannaturali, non è causa dei dilettanti, cialtroni, delinquenti che hanno smontato con determinazione e scientificità lo Stato italiano negli ultimi vent'anni. Quei padri che rifiutano ogni addebito del disastro nazionale, che percepiscono però vitalizi e doppie pensioni, gente canuta che non ha mai avuto il problema della disoccupazione e del pane quotidiano, è ancora qui, ancora a spiegarci come e perché siano le nuove generazioni, i choosy, i bamboccioni, i veri colpevoli. A raccontarci la favola che affidandosi a loro, alla loro esperienza e capacità e senso dello Stato, si cambierà il Paese. Questo dicono i Padri Puttanieri, quelli che hanno sulle spalle la più grande rapina ai danni delle giovani generazioni. Questi padri che chiagnono e fottono sono i Bersani, i D'Alema, i Berlusconi, i Cicchitto, i Monti che ci prendono allegramente per il culo ogni giorno con i loro appelli quotidiani per la governabilità. Hanno governato a turno per vent'anni, hanno curato i loro interessi, smembrato il tessuto industriale, tagliato lo Stato sociale, distrutto l'innovazione e la ricerca. Pdl e pdmenoelle sono vent'anni che ci prendono per il culo e non hanno ancora

il pudore di togliersi in modo spontaneo dai coglioni dopo Penati, Tedesco, Dell'Utri, Cuffaro, Monte Paschi di Siena, dopo il Lodo Alfano, lo Scudo Fiscale e cento leggi abominio. Vent'anni senza riuscire a produrre una legge contro la corruzione e contro il conflitto di interessi, vent'anni per trasformare la legge elettorale in una caricatura anticostituzionale, senza mai trovare il tempo (ah, il tempo...) per cambiarla. I figli di NN vi manderanno a casa, in un modo o nell'altro, il tempo è dalla loro parte. Hanno ricevuto da voi solo promesse e sberleffi, non hanno nulla da perdere, non hanno un lavoro, né una casa, non avranno mai una pensione e non possono neppure immaginare di farsi una famiglia. Vi restituiranno tutto con gli interessi".

Le variabili più importanti che compongono il nostro dataset, e che useremo nelle nostre analisi sono le seguenti:

1. id: indica l'indirizzo id dell'users che ha scritto il commento raccolto;
2. text: testo dei commenti;
3. timestamp: tempo di ricevimento del commento;

Il range in cui sono stati raccolti tutti i commenti dell'intero dataset, va da "giorno: 27-03-2013, ore: 20.11" al " giorno 27-03-2013, ore: 22.53".

In queste analisi viene analizzato l'atteggiamento che le persone assumono nel confronto dell'argomento del blog di Grillo.

L'approccio ReadMe offre una stima della percentuale delle categorie scelte e propone una sintesi del sentimento verso questo argomento.

Nelle prime tre analisi svolte, dall'intera popolazione dei commenti sono stati presi in considerazione dei campioni che contengono 400 commenti. In ogni analisi viene creato un dataset composto dal campione.

Il campione è diviso in training-set, formato da 200 commenti classificati a mano e un test-set formato da altrettanti commenti.

Nell'analisi 4 ci mettiamo artificialmente in una condizione in cui il test-set è molto diverso dal training-set.

In seguito abbiamo analizzato questi dataset utilizzando sia il metodo ReadMe e l'albero di classificazione per poter confrontare le due strategie e trovare quella che al meglio rappresentasse le informazioni raccolte.
Nelle prime tre analisi, la fase di codifica dei commenti (classificazione), svolta a mano, ha creato alcuni problemi.
L'azione di codifica dipende direttamente dal ricercatore, è un azione soggettiva, l'argomento che si tratta non è una variabile quantitativa, ma una variabile molto più complicata da analizzare, legata alle parole usate.
Come riportato precedentemente, si analizza l'atteggiamento delle persone riguardo il post del blog di Grillo, la tendenza che gli utenti assumono e trasmettono tramite i commenti.
Il primo problema è dato dalla scelta delle categorie di codifica. Inizialmente abbiamo assunto 7 categorie divise in questo modo :

-1: atteggiamento negativo sul blog.
-11: atteggiamento negativo su Grillo.
0: atteggiamento neutro.
+1: atteggiamento positivo sul blog.
+11: atteggiamento positivo sul Grillo.
3: atteggiamento positivo per l M5S
4: atteggiamento non attinente con le analisi svolte.

La presenza di tre categorie positive e due negative complica la fase di codifica, dunque si cerca di capire quale di queste è meno influente e meno frequente per restringere il numero delle categorie.
Si è svolta una analisi ristretta su un campione di 100 commenti e si sono calcolate le seguenti frequenze:

	-11	-1	0	1	11	3	4
Freq.	0.25	0.06	0.15	0.08	0.13	0.06	0.27

Tab. 3.1

Si può notare che alcune categorie sono poco influenti all'interno della codifica, in quanto presentano una frequenza troppo bassa, per questo si può pensare di includerle all'interno di altre categorie. Raggruppiamo quindi le categorie +1, +3, +11 e le categorie -1 , -11 poiché contengono sentimenti simili.
Infine le categorie usate sono state:

+1: atteggiamento positivo.
0: atteggiamento neutro.
-1: atteggiamento negativo.
4: atteggiamento non attinente.

Analisi 1

L'analisi 1 è svolta utilizzando un dataset formato da un training-set dei primi 200 commenti classificati a mano e da un test-set composto dagli ultimi 200 commenti. Le classificazioni utilizzate sono formate da 4 categorie (negativo, neutro, positivo e non attinente), ed è stato applicato il metodo ReadMe e l'albero di classificazione per prevedere la classificazione degli ultimi 200 commenti (test-set).
I risultati ottenuti da questi due metodi sono riportati nella seguente tabella:

	Negativo (-1)	Neutro (0)	Positivo (+1)	Non attinente (4)
% Reale	0.291	0.146	0.306	0.256
% ReadMe	0.330	0.114	0.382	0.172
% Albero	0.284	0.171	0.301	0.243

Tab. 3.2. Training-set: i primi 200, test-set: gli ultimi 200.

Utilizziamo un grafico per rappresentare la differenza per ogni categoria, rispettivamente tra la percentuale stimata di ReadMe/albero di classificazione e quella reale:

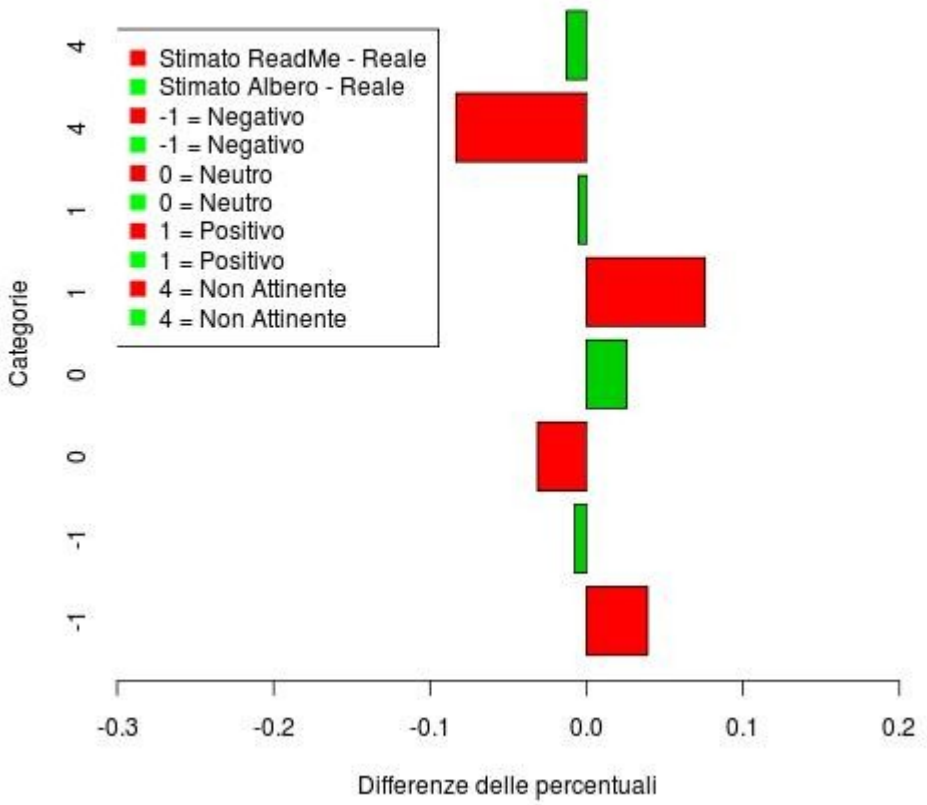

Fig. 3.1

Per rendere più chiari questi dati è possibile utilizzare la somma degli scarti al quadrato, tra stimati e reali.

La somma degli scarti ottenuta tra ReadMe e reali è 0.015 e tra l'albero di classificazione e reali è 0.009; notiamo che in questo caso non c'è differenza nell'usare uno o l'altro metodo, ma per poco potremmo preferire il metodo classificatore ad albero.

Analisi 2

L'analisi 2 è svolta utilizzando un dataset formato da un training-set di 200 commenti presi a caso, classificati a mano e da un test-set composto dai primi 200 commenti. Le classificazioni utilizzate sono formate da 4 categorie (negativo,

neutro, positivo e non attinente), ed è stato applicato il metodo ReadMe e l'albero di classificazione per prevedere la classificazione dei primi 200 commenti (test-set).
I risultati ottenuti da questi due metodi sono riportati nella seguente tabella:

	Negativo (-1)	Neutro (0)	Positivo (+1)	Non attinente (4)
% Reale	0.290	0.145	0.290	0.275
% ReadMe	0.272	0.111	0.349	0.267
% Albero	0.265	0.120	0.334	0.281

Tab. 3.3: Training-set: i 200 presi a caso; test-set: i primi 200

Utilizziamo, anche in questo caso, un grafico per rappresentare la differenza per ogni categoria, rispettivamente tra la percentuale stimata di ReadMe/albero di classificazione e quella reale:

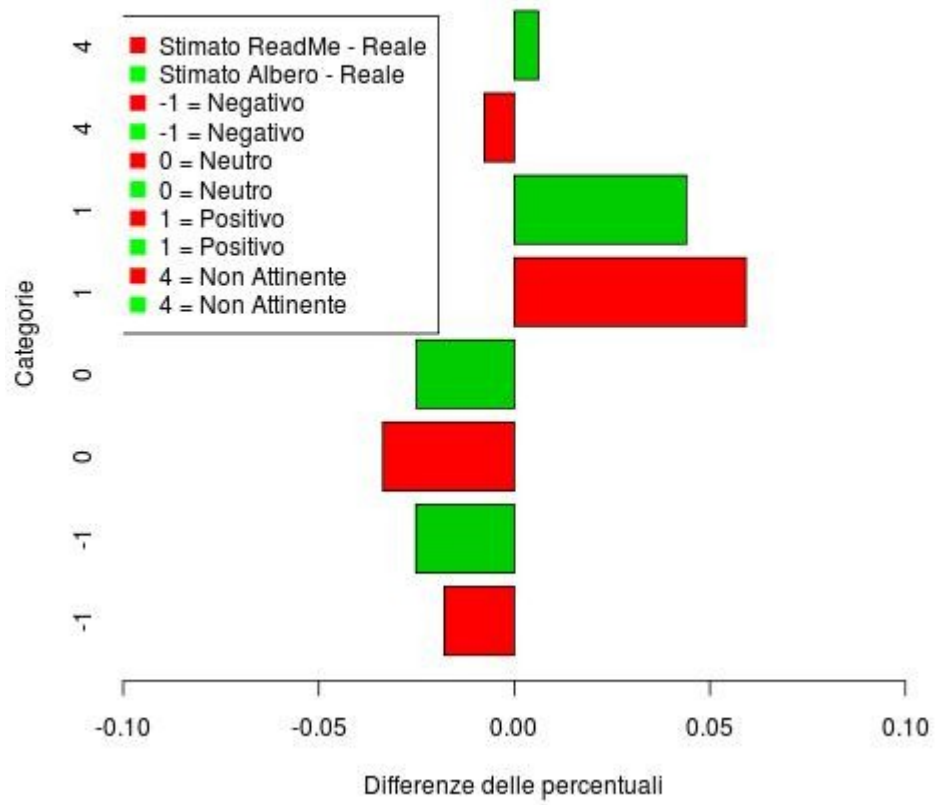

Fig. 3.2

La somma degli scarti al quadrato ottenuta tra ReadMe e reali è 0.005 e tra l'albero di classificazione e reali è 0.003. Anche in questa seconda analisi è possibile notare che non vi è sostanziale differenza tra i due metodi.

Analisi 3

L'analisi 2 è svolta utilizzando un dataset formato da un training-set di 200 commenti presi a caso, classificati a mano e da un test-set composto dagli ultimi 200 commenti. Le classificazioni utilizzate sono formate da 4 categorie (negativo,

neutro, positivo e non attinente), ed è stato applicato il metodo ReadMe e l'albero di classificazione per prevedere la classificazione degli ultimi 200 commenti (test-set).

I risultati ottenuti da questi due metodi sono riportati nella seguente tabella:

	Negativo (-1)	Neutro (0)	Positivo (+1)	Non attinente (4)
% Reale	0.295	0.145	0.305	0.255
% ReadMe	0.358	0.105	0.408	0.128
% Albero	0.298	0.120	0.345	0.237

Tab. 3.4: Training-set: i 200 presi a caso, test-set: 200 ultimi

Utilizziamo, anche in questo caso, un grafico per rappresentare la differenza per ogni categoria, rispettivamente tra la percentuale stimata di ReadMe/albero di classificazione e quella reale:

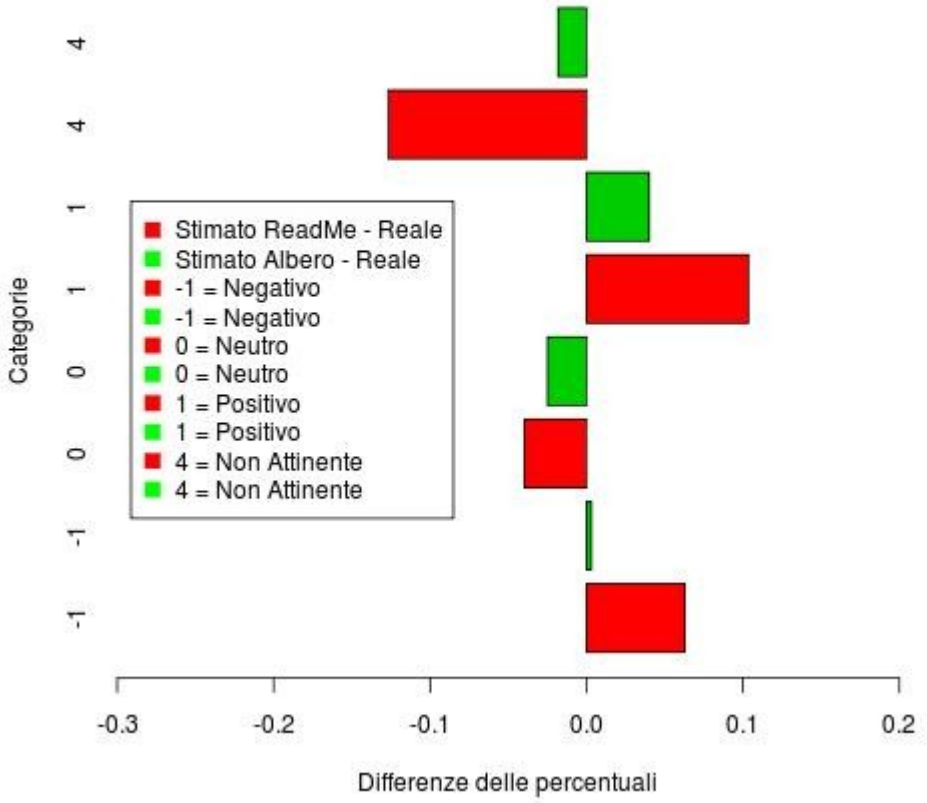

Fig. 3.3

Calcolando la somma degli scarti al quadrato tra ReadMe e reali si ottiene il valore di 0.032, mentre tra l'albero di classificazione e reali tale valore è leggermente inferiore (0.025). Per questo motivo se dovessimo scegliere uno dei due metodi, preferiremmo l'albero di classificazione.

Analisi 4

L'analisi 4 è stata svolta in due casi estremi: abbiamo posto i 200 commenti presi a caso come training-set e come test-set per il primo caso commenti riferiti ai sentimenti negativi e neutri (-1 e 0) e per il secondo caso commenti positivi e

non attinenti (1 e 4).

Ci mettiamo artificialmente in una condizione in cui il test-set è molto diverso dal training-set, cosi è possibile vedere se i due metodi sono in grado di prevedere in modo esatto.

	Negativo (-1)	Neutro (0)	Positivo (+1)	Non attinente (4)
% Reale	0.674	0.326	0.000	0.000
% ReadMe	0.275	0.119	0.375	0.232
% Albero	0.248	0.109	0.360	0.283

Tab. 3.5: Training-set: i 200 messaggi a random, test-set: -1 e 0

	Negativo (-1)	Neutro (0)	Positivo (+1)	Non attinente (4)
% Reale	0.000	0.000	0.527	0.473
% ReadMe	0.310	0.109	0.402	0.178
% Albero	0.229	0.116	0.361	0.293

Tab. 3.6: Training-set: i 200 messaggi a random, test-set: 1 e 4

I risultati di questa analisi non sono quelli attesi. Questi ultimi dovevano essere nel primo caso maggiori nelle colonne “negativo” e “neutro” e tendenti allo zero nelle colonne “positivo” e “non attinente”. Nel secondo caso, in cui il test-set è caratterizzato dalle categorie “positivo” e “non attinente”, ci attenderemo esattamente l'opposto.

Capitolo IV

4.1 Conclusioni

Il nostro studio svolto sul post di un blog di grillo ci ha portato a dare delle considerazioni sull'utilizzo e le prestazioni di ReadMe e l'albero di classificazione.

Riportiamo una tabella con un riassunto delle prime 3 analisi rispetto l'utilizzo dei due metodi.

	Analisi 1	Analisi 2	Analisi 3
Stimati ReadMe - Reali	0.015	0.005	0.032
Stimati Albero - Reali	0.009	0.003	0.025

Tab. 4.1

Osservando questa tabella, con le somme degli scarti al quadrato, possiamo notare che per sia l'analisi 1 (training-set: i primi 200, test-set: ultimi 200), sia per l'analisi 2 (training-set: 200 presi caso, test-set: primi 200) e sia per l'analisi 3 (training-set: 200 presi a caso, test-set: ultimi 200) è indifferente usare sia il metodo Readme che l'albero di classificazione , quindi possiamo concludere che è preferibile non basarsi solo su uno dei due metodi per classificare, ma utilizzarli entrambi o aggiungerne degli altri.

Con l 'analisi 4 possiamo notare che, nel nostro caso, i due metodi non riescono a predire in modo efficiente i commenti, poiché ci si aspettava di avere valori più alti nelle colonne corrispondenti alle categorie considerate come test-set e valori tendenti allo zero nelle restanti categorie (Tab. 3.5 e Tab. 3.6).

Tali risultati inattesi possono essere dovuti ad un deficit dei metodi utilizzati, o agli stessi commenti troppo simili tra di loro. Per chiarire questo dubbio, per ipotesi si potrebbe svolgere la stessa analisi su diversi dataset composti da commenti su argomenti più concisi.

Abbiamo effettuato un altra analisi, che prende in considerazione sia un training-set che un test-set più ampio. Un dataset formato da un training-set di 600 commenti composto dalla somma dei primi 200, gli ultimi 200 e 200 commenti

presi a caso e un test-set formato dai restanti 5997.

Abbiamo provato con questo dataset a vedere se, con un numero maggiore di commenti utilizzati per l'analisi, fosse cambiata l'efficacia dei due metodi.

Nella tabella riportiamo la frequenza del trainig-set e dell'albero di classificazione e di ReadMe:

	Negativo (-1)	Neutro (0)	Positivo (+1)	Non attinente (4)
Freq. traing-set	0.283	0.137	0.302	0.278
% ReadMe	0.222	0.151	0.253	0.373
% Albero	0.286	0.137	0.298	0.280

Tab. 4.2

In questo caso emerge chiaramente che l'albero di classificazione prevede esattamente come fosse il trainig-set.

Concludendo, seppure con i limiti descritti, i due metodi utilizzati rappresentano un importante strumento per l'analisi di post presenti nei vari social network, i quali sono sempre più sfruttati come mezzo di comunicazione tra aziende e privati.

Un miglioramento e un più largo utilizzo di tali metodi porterà in futuro, un contributo allo sviluppo della Sentiment Analysis, uno degli strumenti più usati per determinare il legame e i sentimenti espressi dalla rete.

APPENDICE A

Funzione ReadMe

A.1 Funzione Undergrad

Questa funzione traduce una serie di documenti di testo memorizzati in una singola cartella in un insieme dove ogni testo è rappresentato da una riga e ogni parola da una colonna.
La sintassi:

Undergrad(control=”control.txt”,stem=T,
strip.tags=T,ignore.case=T,table.file=”tablefile.txt”,
threshold=0,01,python3=F,pyexe=NULL,sep=NULL,printit=TRUE)

in questa tabella si possono vedere gli input della funzione:

Control	Specificare il file da caricare.
Stem	Di default è settato a True.
Strip.tags	Settato a True elimina i tag HTML / XML / SGML.
Table.file	Percorso del file in cui la tabella delle frequenze delle parole dovrebbe essere salvata. Di default è “tablefile.txt”.
Threshold	È un numero compreso tra 0 e 1 ed indica una percentuale, verranno incluse solo le parole che superano questa soglia. Per includere tutte le parole settiamo threshold a 0. Di default e pari a 0.01 che equivale all'1%.
Pyexe	Percorso da utilizzare per l'interprete Python. Se impostato a Null, ReadMe cercherà prima nel percorso di sistema e quindi su Windows o linux nella directory di installazione di default. Se ReadMe è in grado di individuare il vostro interprete Python o si desidera utilizzare un interprete diverso rispetto a quello che si trova sul percorso di sistema impostare questa variabile. Di default è impostato a Null.
Python3	Impostare a True se si utilizza python

	3.0 di default è False.
Sep	Indicre il eparatore che si utilizza per separare le colonne in “ontrol.txt”. Di default è Null.
Printit	Variabile booleana che indica se si vuole vedere a schermo lo sviluppo del processo. Di default è impostato a True.
Fullfreq	Variabile booleana che indica se impostata a True la frequenza delle parole oppure se impostata a False restituisce un numero binario che indica la presenza o l'assenza della parola.

Gli output della frequenza:

Triningset	Tabella binaria che indica quali parole, tra quelle che soddisfano la clausola threshold, appaiono in ogni testo del training-set.
test-set	Stesso formato del training-set, ma per il test-set.
Formula	Formula da utilizzare nella chiamat alla funzione VA. Scritta secondo lo standard R. di default include tutte le parole che soddisfano la clausola threshold e appaiono meno del 100%, cioè sono parole costanti e vengono identificate come variabili indipendenti, “truth” dal file control sono le variabili esplicative.
Features	Numero di caratteristiche da usare in ogni sottoinsieme in VA. Corrispondente a nsymp in VA. Di default è uguale 15.
N.subset	Numero di sottoinsieme da usare in VA. Valore di default è 300. numeri elevati producono stime più precise.
Prob.wt	Vettore di probabilità pesate per le caratteristiche per essere impiegate da VA. Dovrebbe avere lunghezza uguale al numero di caratteristiche nella formula. Di default è uguale a 1.
Boot.se	Calcolo errore dello standard error via

	bootstrap. Default impostato a False.
Nboot	Numero di ricampionamenti da fare via bootstrap. Di default impostato a 300.
Printit	Stampa a video oppure no il progresso della funzione VA. Di default impostato a True.

A.2 Funzione PREPROCESS

quenza funzione prende in input la matrice dei dati creata dalla funzione undergrad() e prepara essa per l'analisi di ReadMe() rimuovendo le colonne invarianti.
Sintassi:

preprocess(undergrad.results)

Si vedono gli input:

undergrad.results	L'output da ReadMe.

Si vedono gli output:

Undergrad.preprocessed	Una lista pre elaborata con colonne costanti rimosse.

A.3Funzione ReadMe

Questa funzione calcola la percentuale di documenti di testo all'interno di ogni categoria specificata dall'utente.
ReadMe richiede in R la funzione VA.
Sintassi:

ReadMe(undergradlist = list(), training-set = Null,test-set = Null, formula = Null, n.subset = Null, prob.wt = Null, boot.se = Null, nboot = Null, printit = Null)

Vengono riportati i parametri di ingresso alla funzione:

Undergradlist	Una lista che ha un elemento per ogni parametro. Specificare i valori dei parametri o attraverso una lista o sttraverso singoli elementi. Singoli argomenti non nulli sostituiscono valori alla lista.

Features	Numero intero positivo che specifica il numero di parole per sottoinsieme da tutte le parole per stimare ogni iterazione. Per la scelta delle features. Di default è impoistato a 16.
Formula	La formula specifica tutte le possibili caratteristiche (vedi funzione undergrad()). Per modificarla la nuova formula deve essere definita come: formula=cbind(word.the+word.formula)~ TRUTH
n.subset	Numeropositivo intero, specifica il numero totale di campioni di differenti sottoinsiemi di caratteristiche. Di default è impostato a 300.
Prob.wt	Numero intero positivo o un vettore di pesi che determinano le probabilità che un elemento dee avere se si seleziona il sottoinsieme di caratteristiche. Quando prob.wt è un , la sua dimensione è uguale al numero di caratteristiche e le probabilità sommate danno 1, quando prob.wt= 0 tutte le caratteristiche saranno selezione in modo equiprobabile, quando prob.wt=1 pesi binomiali la quale sono proporzioni.
Boot.se	Valore logico, se impostato a True, il botstrap standard errors di CSMF sono stimati. Questa opzione richiede generalmente molto tempo per l'esecuzione. Di default è impostato a False.
Nboot	Numero intero positivo, se boot.se=True, esso spwcifica il numero di campioni da prendere per stimare lo standard errors di CSMF di default assume il valore 300.
Printit	Valore logico, se impostato a True il progresso delle stime viene stampato sullo schermo.

Vengono riportati i parametri dell output della funzione:

Est.CSMF	La stima della percentuale di ogni categoria.

True.CSMF	Le percentuali osservate in ogni categoria, quando è disponibile.
Est.se	Il bootstrap standard errors di est.CSMF quando boot.se=True
True.CSMF.bootmean	La media delle percentuali osservate per ogni categoria via bootstrap quando sono disponibili o quando boot.se=True
True.bootse	Standard errors delle percentuali osservate per ogni categoria via bootstrap quando sono disponibili o quando boot.se=True.

APPENDICE B

B.1 Script per la creazione ed analisi per un traing-set formato dai primi 200 e test-set formato dagli ultimi 200 commenti del blog di Grillo

```
#creiamo il campione
x=200
y=200
#si crea il campione con i primi 200 commenti
idGrillo=grep(" ?",(db$text))
#idGrillo
#creiamo un database con solo i dati relativi ai primi 200 commenti
dbGrilloUlt=db[idGrillo[1:200],]
dbGrilloPri=db[idGrillo[6398:6597],]
dbGrillo
head(dbGrillo)
sampleGrilloPri=sample(nrow(dbGrilloPri),x)
sampleGrilloUltTutto=sample(nrow(dbGrilloUlt),y)
dbGrillo$text[sampleGrillo]
#si cambia cartella
setwd("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr")

#verifichiamo di essere all'interno della cartella che vogliamo/
getwd()
#si crea tuttii singoli file di testo per i primi  200
sapply(sampleGrilloPri,function(id)cat(dbGrilloPri$text[id],file=paste(id,sep="","
a.txt")))
#si crea tutti i singoli file per gli ultimi 200
sapply(sampleGrilloUlt,function(id)cat(dbGrilloUlt$text[id],file=paste(id,sep="","
.txt")))
#creo un file di testo che contiene tutti ifile di testo creati in precedenza, questo
```

file è denominato controlPre.txt

```
write.csv(file="controlPrePri.txt",cbind(FILENAME=paste(sampleGrilloPri,"a.txt
",sep=""),TRUTH=NA,TRAININGSET=rep(1,200),message=dbGrilloPri$text[sa
mpleGrilloPri]),row.names=FALSE)
write.csv(file="controlPreUlt.txt",cbind(FILENAME=paste(sampleGrilloUlt,".txt
",sep=""),TRUTH=NA,TRAININGSET=rep(0,200),message=dbGrilloUlt$text[sa
mpleGrilloUlt]),row.names=FALSE)
#leggo il file di testo controlPre.txt appena classificato//
controlPrePri=read.csv("/home/giulia/R/i486-pc-linux-gnu-
library/2.10/ReadMe/GrilloConfr/controlPrePri.txt")
controlPreUlt=read.csv("/home/giulia/R/i486-pc-linux-gnu-
library/2.10/ReadMe/GrilloConfr/controlUlt.txt")
controlTot=rbind(controlPrePri,controlPreUlt)
setwd("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr")
write.csv(file="controlPre.csv",controlTot)
testoUltimi=read.csv("/home/giulia/R/i486-pc-linux-gnu-
library/2.10/ReadMe/GrilloConfr/controlPre.csv",stringsAsFactors=FALSE)
str(testoUltimi)
attach(testoUltimi)
#creo un dataframe che prende solo le colonne filename , truth , traininngset
control=data.frame(FILENAME,TRUTH,TRAININGSET)
write.table(control,file="control.txt",row.names=FALSE,sep=",",quote=FALSE)
#ora creo il file tableFile.txt
library(ReadMe)
library(quadprog)
```

#traduce una serie di documenti di testo memorizzati in una singola cartella in un insieme, dove ogni testo è rappresentato da una riga e ogni parola da una colonna . Sep indica il separatore che si utilizza per separare le colonne di default è Null.

```
undergrad.results<-undergrad(control="control.txt",sep=",")
```

#questa funzione prende in input la matrice dei dati creata dalla funzione undergrad()

```
undergrad.preprocess<-preprocess(undergrad.results)
undergrad.preprocess
#si mostrano quali colonne sono state eliminate
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),lengt
h(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))
#calcola la percentuale di documenti di testo all'interno di ogni categoria
specificata.
readme.results<-readme(undergrad.preprocess)
readme.results
#stima percentuale di ogni categoria
readme.results$est.CSMF
#reale percentuale per ogni categoria
readme.results$true.CSMF
#albero di classificazione
verita=undergrad.preprocess[["trainingset"]][["TRUTH"]]
verita=as.factor(verita)
verita
stem=(undergrad.preprocess[["trainingset"]][,c(1,2,3)])
x=data.frame(undergrad.preprocess[["trainingset"]][,-c(1,2,3)],verita)
attach(x)
head(x)
#caricare pacchetto rpart
library(rpart)
albero=rpart(verita~.,data=x,method="class")
summary(albero)
albero
print(albero)
#per migliorare la grafica, che le scritte non vengano tagliate
par(mar=c(1,1,1,1),xpd=TRUE)
#si visualizza l'albero
plot(albero)
si aggiungo le condizioni ai nodi
```

```
text(albero,all=TRUE,use.n=TRUE)
#stima dell'errore
#il criterio è allora selezionare l'albero che realizza il minimo xerror o quello corrispondente al più grande valore di xerror minore della somma del minimo dell'errore standard (xstd).
par=(mar=c(5,4,4,1))
plotcp(albero)
#serve per analizzare i risultati,la tabellina che viene stampata riporta, per ciascuno gradi di complessità(misurato dal numero di suddivisioni nsplit,pari al numero di foglie meno uno),
#due misure d'errore espresse relativamente all'errore del nodo radice (cioè all'errore della regola diclassificazione che assegna tutte le unità
#alle classepiù numerose):rel error è l'errore apparente, e xerror è l'errore valutato mediante validazione incrociata, l'ultima colonna è una stima della variabilità della stima d'errore.
printcp(albero)
#stima del testSet
new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])
head(new.stem)
#per saper come potare l'albero
tabella=printcp(albero)
cp.min=tabella[which.min(tabella[,4]),1]
cp.min
#cp=.... si sceglie inconseguenza al cp.min
potature=prune(albero,cp=0.09)
#la previsione sull'albero potato si ottiene
predizione.probabilità=predict(albero,newdata=new.stem)
predizione.probabilità
media=colMeans(predizione.probabilità)
media
#somma degli scarti al quadrato
predetti=readme.results$est.CSMF
```

```
reali=readme.results$true.CSMF
scartiReadMe
scartiReadMe
predetti
reali
#costruzione del grafivo per vedere la differenza tra ReadMe e l'albero di classificazione
barplot(c(predetti[1]-reali[1], media[1]-reali[1],predetti[2]-reali[2], media[2]-reali[2],predetti[3]-reali[3], media[3]-reali[3],predetti[4]-reali[4], media[4]-reali[4]), horiz=TRUE, beside=TRUE, col=c(2,3), ylab="Categorie", xlab="Differenze delle percentuali", xlim=c(-0.30,+0.20),offset=0)
#per posizionare la leggenda
a=locator(1)
a
#costruzione leggenda
legend(a,legend=c("Stimato ReadMe - Reale", "Stimato Albero - Reale", "-1 = Negativo", "-1 = Negativo", "0 = Neutro", "0 = Neutro","1 = Positivo", "1 = Positivo", "4 = Non Attinente", "4 = Non Attinente"), col=c("red", "green"), pch=22, pt.bg=c("red", "green"), pt.cex=1.5)
```

B.2 Script per creazione ed analisi per un traing-set formato dai 200 commenti presi a caso e un test-set formato dai primi 200 commenti

```
#creiamo il campione
x=200
y=200
#si crea il campione con i primi 200 commenti
idGrillo=grep(" ?",(db$text))
#idGrillo
#creiamo un database con solo i dati relativi ai primi 200 commenti
dbGrilloUlt=db[idGrillo[1:200],]
dbGrilloPri=db[idGrillo[6398:6597],]
```

```
dbGrillo
head(dbGrillo)
sampleGrilloPri=sample(nrow(dbGrilloPri),x)
sampleGrilloUltTutto=sample(nrow(dbGrilloUlt),y)
dbGrillo$text[sampleGrillo]
#si cambia cartella
setwd("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr3")
#verifichiamo di essere all'interno della cartella che vogliamo/
getwd()
#si crea tuttii singoli file di testo
sapply(sampleGrilloCao,function(id)cat(dbGrilloPri$text[id],file=paste(id,sep="",
"a.txt")))
sapply(sampleGrilloPri,function(id)cat(dbGrilloUlt$text[id],file=paste(id,sep="","
.txt")))
#creo un file di testo che contiene tutti ifile di testo creati in precedenza, questo
file è denominato controlPre.txt
write.csv(file="controlPreCao.txt",cbind(FILENAME=paste(sampleGrilloCao,"a.t
xt",sep=""),TRUTH=NA,TRAININGSET=rep(1,200),message=dbGrilloCao$text
[sampleGrilloCa0]),row.names=FALSE)
write.csv(file="controlPrePri.txt",cbind(FILENAME=paste(sampleGrilloPri,".txt"
,sep=""),TRUTH=NA,TRAININGSET=rep(0,200),message=dbGrilloPri$text[sa
mpleGrilloPri]),row.names=FALSE)
#leggo il file di testo controlPre.txt appena classificato//
controlPreCao=read.csv("/home/giulia/R/i486-pc-linux-gnu-
library/2.10/ReadMe/GrilloConfr3/controlPreCao.txt")
controlPrePri=read.csv("/home/giulia/R/i486-pc-linux-gnu-
library/2.10/ReadMe/GrilloConfr3/controlPrePri.txt")
controlTot=rbind(controlPreCao,controlPrePri)
setwd("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr3")
write.csv(file="controlPre.csv",controlTot)
testoUltimi=read.csv("/home/giulia/R/i486-pc-linux-gnu-
library/2.10/ReadMe/GrilloConfr3/controlPre.csv",stringsAsFactors=FALSE)
```

```
str(testoUltimi)
attach(testoUltimi)
#creo un dataframe che prende solo le colonne filename , truth , traininngset
control=data.frame(FILENAME,TRUTH,TRAININGSET)
write.table(control,file="control.txt",row.names=FALSE,sep=",",quote=FALSE)
#ora creo il file tableFile.txt
library(ReadMe)
library(quadprog)
#traduce una serie di documenti di testo memorizzati in una singola cartella in un insieme, dove ogni testo è rappresentato da una riga  e ogni parola da una colonna . Sep indica il separatore che si utilizza per separare le colonne di default è Null.
undergrad.results<-undergrad(control="control.txt",sep=",")
#questa funzione prende in input la matrice dei dati creata dalla funzione undergrad()
undergrad.preprocess<-preprocess(undergrad.results)
undergrad.preprocess
#si mostrano quali colonne sono state eliminate
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),length(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))
#calcola la percentuale di documenti di testo all'interno di ogni categoria specificata.
readme.results<-readme(undergrad.preprocess)
readme.results
#stima percentuale di ogni categoria
readme.results$est.CSMF
#reale percentuale per ogni categoria
readme.results$true.CSMF
#albero di classificazione
verita=undergrad.preprocess[["trainingset"]][["TRUTH"]]
verita=as.factor(verita)
```

```
verita
stem=(undergrad.preprocess[["trainingset"]][,c(1,2,3)])
x=data.frame(undergrad.preprocess[["trainingset"]][,-c(1,2,3)],verita)
attach(x)
head(x)
#caricare pacchetto rpart
library(rpart)
albero=rpart(verita~.,data=x,method="class")
summary(albero)
albero
print(albero)
#per migliorare la grafica, che le scritte non vengano tagliate
par(mar=c(1,1,1,1),xpd=TRUE)
#si visualizza l'albero
plot(albero)
si aggiungo le condizioni ai nodi
text(albero,all=TRUE,use.n=TRUE)
#stima dell'errore
#il criterio è allora selezionare l'albero che realizza il minimo xerror o quello corrispondente al più grande valore di xerror minore della somma del minimo dell'errore standard (xstd).
par=(mar=c(5,4,4,1))
plotcp(albero)
#serve per analizzare i risultati,la tabellina che viene stampata riporta, per ciascuno gradi di complessità(misurato dal numero di suddivisioni nsplit,pari al numero di foglie meno uno),
#due misure d'errore espresse relativamente all'errore del nodo radice (cioè all'errore della regola diclassificazione che assegna tutte le unità
#alle classepiù numerose):rel error è l'errore apparente, e xerror è l'errore valutato mediante validazione incrociata, l'ultima colonna è una stima della variabilità della stima d'errore.
printcp(albero)
```

```
#stima del testSet
new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])
head(new.stem)
#per saper come potare l'albero
tabella=printcp(albero)
cp.min=tabella[which.min(tabella[,4]),1]
cp.min
#cp=.... si sceglie inconseguenza al cp.min
potature=prune(albero,cp=0.09)
#la previsione sull'albero potato si ottiene
predizione.probabilità=predict(albero,newdata=new.stem)
predizione.probabilità
media=colMeans(predizione.probabilità)
media
#somma degli scarti al quadrato
predetti=readme.results$est.CSMF
reali=readme.results$true.CSMF
scartiReadMe
scartiReadMe
predetti
reali
#costruzione del grafivo per vedere la differenza tra ReadMe e l'albero di classificazione
barplot(c(predetti[1]-reali[1], media[1]-reali[1],predetti[2]-reali[2], media[2]-reali[2],predetti[3]-reali[3], media[3]-reali[3],predetti[4]-reali[4], media[4]-reali[4]), horiz=TRUE, beside=TRUE, col=c(2,3), ylab="Categorie", xlab="Differenze delle percentuali", xlim=c(-0.30,+0.20),offset=0)
#per posizionare la leggenda
a=locator(1)
a
#costruzione leggenda
legend(a,legend=c("Stimato ReadMe - Reale", "Stimato Albero - Reale", "-1 =
```

Negativo", "-1 = Negativo", "0 = Neutro", "0 = Neutro","1 = Positivo", "1 = Positivo", "4 = Non Attinente", "4 = Non Attinente"), col=c("red", "green"), pch=22, pt.bg=c("red", "green"), pt.cex=1.5)

B.3 Script per creazione ed analisi formato il traing-set con i 200 commenti presi a caso e il test-set formato dagli ultimi 200 commenti

```
#creiamo il campione
x=200
y=200
#si crea il campione con i primi 100 commenti
idGrillo=grep(" ?",(db$text))

#idGrillo
#creiamo un database con solo i dati relativi ai primi 100 commenti
dbGrilloUlt=db[idGrillo[1:200],]
dbGrilloPri=db[idGrillo[6398:6597],]
dbGrillo
head(dbGrillo)
sampleGrilloPri=sample(nrow(dbGrilloPri),x)
sampleGrilloUltTutto=sample(nrow(dbGrilloUlt),y)
dbGrillo$text[sampleGrillo]
#si cambia cartella
setwd("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr2")

#verifichiamo di essere all'interno della cartella che vogliamo/
getwd()
#si crea tuttii singoli file di testo
sapply(sampleGrilloPri,function(id)cat(dbGrilloPri$text[id],file=paste(id,sep="","
a.txt")))
sapply(sampleGrilloUlt,function(id)cat(dbGrilloUlt$text[id],file=paste(id,sep="","
.txt")))
```

#creo un file di testo che contiene tutti ifile di testo creati in precedenza, questo file è denominato controlPre.txt

write.csv(file="controlPrePri.txt",cbind(FILENAME=paste(sampleGrilloPri,"a.txt",sep=""),TRUTH=NA,TRAININGSET=rep(1,200),message=dbGrilloPri$text[sampleGrilloPri]),row.names=FALSE)

write.csv(file="controlPreUlt.txt",cbind(FILENAME=paste(sampleGrilloUlt,".txt",sep=""),TRUTH=NA,TRAININGSET=rep(0,200),message=dbGrilloUlt$text[sampleGrilloUlt]),row.names=FALSE)

#leggo il file di testo controlPre.txt appena classificato//

controlPreCao=read.csv("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr2/controlPreCao.txt")

controlPreUlt=read.csv("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr2/controlPreUlt.txt")

controlTot=rbind(controlPreCao,controlPreUlt)

setwd("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr2")

write.csv(file="controlPre.csv",controlTot)

testoUltimi=read.csv("/home/giulia/R/i486-pc-linux-gnu-library/2.10/ReadMe/GrilloConfr2/controlPre.csv",stringsAsFactors=FALSE)

str(testoUltimi)

attach(testoUltimi)

#creo un dataframe che prende solo le colonne filename , truth , traininngset

control=data.frame(FILENAME,TRUTH,TRAININGSET)

write.table(control,file="control.txt",row.names=FALSE,sep=",",quote=FALSE)

#ora creo il file tableFile.txt

library(ReadMe)

library(quadprog)

#traduce una serie di documenti di testo memorizzati in una singola cartella in un insieme, dove ogni testo è rappresentato da una riga e ogni parola da una colonna . Sep indica il separatore che si utilizza per separare le colonne di default è Null.

undergrad.results<-undergrad(control="control.txt",sep=",")

```
#questa funzione prende in input la matrice dei dati creata dalla funzione undergrad()
undergrad.preprocess<-preprocess(undergrad.results)
undergrad.preprocess
#si mostrano quali colonne sono state eliminate
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),length(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))
#calcola la percentuale di documenti di testo all'interno di ogni categoria specificata.
readme.results<-readme(undergrad.preprocess)
#str(undergrad.preprocess)
readme.results
#stima percentuale di ogni categoria
readme.results$est.CSMF
#reale percentuale per ogni categoria
readme.results$true.CSMF
#albero di classificazione
verita=undergrad.preprocess[["trainingset"]][["TRUTH"]]
verita=as.factor(verita)
verita
stem=(undergrad.preprocess[["trainingset"]][,c(1,2,3)])
x=data.frame(undergrad.preprocess[["trainingset"]][,-c(1,2,3)],verita)
attach(x)
head(x)
#caricare pacchetto rpart
library(rpart)
albero=rpart(verita~.,data=x,method="class")
summary(albero)
albero
print(albero)
#per migliorare la grafica, che le scritte non vengano tagliate
par(mar=c(1,1,1,1),xpd=TRUE)
```

#si visualizza l'albero
plot(albero)
si aggiungo le condizioni ai nodi
text(albero,all=TRUE,use.n=TRUE)
#stima dell'errore
#il criterio è allora selezionare l'albero che realizza il minimo xerror o quello corrispondente al più grande valore di xerror minore della somma del minimo dell'errore standard (xstd).
par=(mar=c(5,4,4,1))
plotcp(albero)
#serve per analizzare i risultati,la tabellina che viene stampata riporta, per ciascuno gradi di complessità(misurato dal numero di suddivisioni nsplit,pari al numero di foglie meno uno),
#due misure d'errore espresse relativamente all'errore del nodo radice (cioè all'errore della regola diclassificazione che assegna tutte le unità
#alle classepiù numerose):rel error è l'errore apparente, e xerror è l'errore valutato mediante validazione incrociata, l'ultima colonna è una stima della variabilità della stima d'errore.
printcp(albero)
#stima del testSet
new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])
head(new.stem)
#per saper come potare l'albero
tabella=printcp(albero)
cp.min=tabella[which.min(tabella[,4]),1]
cp.min
#cp=.... si sceglie inconseguenza al cp.min
potature=prune(albero,cp=0.09)
#la previsione sull'albero potato si ottiene
predizione.probabilità=predict(albero,newdata=new.stem)
predizione.probabilità
media=colMeans(predizione.probabilità)

```
media
#somma degli scarti al quadrato
predetti=readme.results$est.CSMF
reali=readme.results$true.CSMF
scartiReadMe
scartiReadMe
predetti
reali
#costruzione del grafivo per vedere la differenza tra ReadMe e l'albero di classificazione
barplot(c(predetti[1]-reali[1], media[1]-reali[1],predetti[2]-reali[2], media[2]-reali[2],predetti[3]-reali[3], media[3]-reali[3],predetti[4]-reali[4], media[4]-reali[4]), horiz=TRUE, beside=TRUE, col=c(2,3), ylab="Categorie", xlab="Differenze delle percentuali", xlim=c(-0.30,+0.20),offset=0)
#per posizionare la leggenda
a=locator(1)
a
#costruzione leggenda
legend(a,legend=c("Stimato ReadMe - Reale", "Stimato Albero - Reale", "-1 = Negativo", "-1 = Negativo", "0 = Neutro", "0 = Neutro","1 = Positivo", "1 = Positivo", "4 = Non Attinente", "4 = Non Attinente"), col=c("red", "green"), pch=22, pt.bg=c("red", "green"), pt.cex=1.5)
```

BIBLIOGRAFIA

- J. Hopkings Daniel, King Gary. 2010. “Amethod of Automated Nonparametric Content Analysis Content Analysis for Social Science”, American Journal of Political Science 54, no. 1:229-247.

- J. Hopkings Daniel, King Gary, Knowles Matthew e Melendez Steven “ReadMe: Software for Automated Content Analysis.”

- Progetto di ricerca dell'università di milano che studia la tecnologia per la Sentiment Analysis http://voicefromtheblogs.com/.

- http://www.eventreport.it/stories/Mercato/84570_social_media_le_aziende_italiane_non_sanno_ancora_usarli_per_fare_business/.

- Carlo Mazzoco,”Le Pmi possono sfruttarla”,articolo preso dal sito internet, 24/01/13

- Articoli presi da Archivio da Repubblica “www.repubblica.it”.

- Chiogna Monica, Pauli Francesco “TECNICHE STATISTICHE DI CLASSIFICAZIONE”, appunti per il corso, 2008.

- Masarotto Guido, M Iacus Stefano, McGraw-Hill “Laboratorio di statistica con R”, 2007. ISBN 9788838663697

- Articoli presi da Archivio del sole 24 ore “www.ilsole24ore.com”.

- Il sito del blog di Grillo www.beppegrillo.it/2013/03/i_figli_di_nn.html/.

- Tesi di laurea di Angela Andreela “Sentiment Analysis e Social Network: il caso della campagna elettorale italiana”