Università degli Studi di Padova
Corso di Laurea in Statistica e Tecnologie Informatiche

Metodo Hopkins-King per la Sentiment Analysis: una valutazione basata sui tweets della campagna elettorale.

Relatore: Dott. LIVIO FINOS
Dipartimento di Scienze Statistiche

Co-relatore: Dott. DARIO SOLARI

Laureando: FABIO CACCO

Anno Accademico 2012/2013

Indice:

Introduzione

La crescente disponibilità di testi digitalizzati presenta enormi opportunità per gli scienziati sociali. La codifica manuale di molti blog, discorsi, documenti governativi, giornali, o altre fonti di testo non strutturato risulta inattuabile.

Gli scienziati informatici hanno metodi per l'analisi automatica del contenuto, la maggior parte sono ottimizzati per classificare singoli documenti. Essi normalmente utilizzano metodi di conteggio delle parole, all'interno di un documento, affiancandoci un data base della "conoscenza", che tengono aggiornato periodicamente. Questo approccio porta a trascurare il senso della frase.

Gli scienziati sociali invece vogliono generalizzare la popolazione di documenti senza trascurare il senso della frase, cercando comunque di classificarli correttamente assegnandoli una categoria ben precisa.

Ci troviamo così di fronte a un paradosso, da un lato abbiamo un insieme potenzialmente vastissimo di argomenti di discussione da cui nasce un interesse per cercare di mappare e monitorare tale varietà di informazioni, dall'altro ci scontriamo con l'inadeguatezza delle tecniche, classiche, per la raccolta dati.

Una tecnica classica come quella dei sondaggi è costosa da progettare e da implementare, è per definizione statica (coglie lo stato di opinioni su un dato argomento in uno specifico spazio temporale), e non ammette errori (se tra il momento della progettazione del sondaggio e la sua implementazione emerge qualche fatto nuovo, il cui peso era stato trascurato, l'unico modo per tenerne conto è rifare l'intera progettazione del sondaggio). In più, i soggetti intervistati rispondono spesso in modo "strategico", mentendo o distorcendo la propria reale opinione, in base a criteri che l'intervistatore non è in grado di controllare.

"Utilizzare un sondaggio tradizionale per catturare l'opinione del web e della blogsfera su un certo tema è come cercare di fare una fotografia a un treno che passa a tutta velocità e che cambia forma proprio mentre ci passa accanto. Una istantanea in questo caso non basta. Occorre al contrario cambiare tecnologia di partenza, e passare dalla fotografia a un vero e proprio film in streaming."[1]

Da questa idea è derivata la necessità di stimare il gradimento o la soddisfazione rispetto ad una opinione, un pensiero o un sentimento, espressi in linguaggio naturale, inerenti ad un qualche argomento di discussione.

La Sentiment Analysis permette di fare tutto questo, applicando metodi per l'elaborazione del linguaggio naturale riuscendo cosi ad estrarre le opinioni.

Essa può essere applicata ai social network che dispongono del servizio di microblogging come

1 Cfr: "Voices from the Blogs"

Twitter e Facebook. Il notevole sviluppo di questi nuovi mezzi di comunicazione utilizzati dall'utente anche per esprimere le proprie opinioni ed essere lette dal popolo del web, ci ha portato negli ultimi anni ad interessarci a cosa pensa la gente.

Twitter, in particolare, è un servizio gratuito del social network che fornisce agli utenti registrati una pagina personale aggiornabile con messaggi di massimo 140 caratteri e dalla sintassi poco complessa.

La Sentiment Analyis offre vastissime opportunità nei più svariati campi.

Ad esempio pensiamo ad una qualsiasi società di commercializzazione di qualsiasi settore che intende analizzare l'approccio di mercato di un nuovo prodotto nei confronti della sua clientela, o ad un personaggio pubblico che vuole entrare o sia già entrato in politica al quale interessa conoscere che tipo di attenzione può suscitare nel suo probabile elettorato al fine di affinare una eventuale strategia in pubblico, lo scopo, in entrambi i casi, è aumentare il consenso.

Si pensi che il 25% delle aziende usa contemporaneamente Facebook, Twitter e Youtube per le attività di comunicazione e per interagire con i propri clienti.

Il canale più utilizzato è Twitter: il 77% delle aziende del mondo ne possiede infatti un account.

Nel secondo capitolo tratteremo il metodo di analisi e monitoraggio di Twitter proposto da Gary King e Hopkins Daniel.J. Attraverso questo metodo siamo in grado di ricostruire l’orientamento di chi scrive su internet. Classificando manualmente un piccolo campione di tweet, (training set, campione etichettato), non necessariamente rappresentativo dell’intera popolazione, il software offre una misurazione delle discussioni in corso nello spazio pubblico permettendo di stimare in automatico e in modo ottimale la distribuzione delle opinioni espresse dai blogger riguardante il tema di ricerca.

I vantaggi della Sentiment Analysis sono: la dinamicità del metodo, che permette di catturare l’opinione della rete e i suoi cambiamenti, in tempo reale e in modo costante nel tempo, la flessibilità, nel senso che può essere adeguata alle esigenze di chi la utilizza e infine la possibilità di attuare un approccio bottom-up (dallo specifico al generale), ciò a differenza dei tradizionali sondaggi. Ci interessiamo delle opinioni di chi scrive in rete, lasciando a loro decidere di quale tema parlare e di come parlarne. Il metodo permette di catturare l’opinione dell’intero universo italiano presente nel web e non di un suo piccolo sotto-insieme, come invece si avverrebbe nel caso di un sondaggio.

Nel terzo capitolo analizzeremo i tweets raccolti durante lo scontro televisivo tra i cinque che ha preceduto le primarie del centro sinistra, avvenute a novembre 2012, ed a questi applicheremo il metodo, proposto da Gary King e Hopkins Daniel.J, e poi confronteremo i risultati ottenuti con un metodo più classico.

CAPITOLO I

1.1 Notazione e Variabili

Il nostro obbiettivo sarà di fare inferenza sull'intera popolazione dei tweets riferiti al solo scontro televisivo e non a tutta la campagna. Per fare ciò dall'intera popolazione estraiamo una popolazione di riferimento e da questa estraiamo un campione. Il campione di tweets verrà diviso in 2 parti, training-set e test-set.

Il primo insieme training-set o definito anche insieme etichettato viene classificato a mano. Ogni documento i, (i = 1,...,n), appartenente a questo viene etichettato assegnandoli una data classe di appartenenza.

La classe del documento viene indicata con la variabile D_i, un singolo documento può assumere un solo valore $D_i=j$, tra i J valori possibili, ($J = 1, \dots, j$).

Il secondo insieme test-set, è la popolazione di tutti documenti di testo che vogliamo classificare, dove ogni documento l, ($l = 1, \dots, L$), verrà classificato utilizzando l'insieme precedente.

Altre variabili vengono definite dagli stessi documenti di testo che appartengono al primo insieme.

Queste variabili servono per definire la presenza o l'assenza di una determinata Word-Stem k, ($k=1,...,K$). Esse vengono indicate con S_{ik}, dove il pedice i indica il documento i-esimo e k la word-stem k-esima. S_{ik} la possiamo considerare come una variabile dicotomica che assume i valori 1 o 0:

$$\boldsymbol{S_{ik}} = \begin{cases} 1 & se\, k\ presente\, nel\, documneto\, i \\ 0 & altrimenti \end{cases} \quad .$$

Allora possiamo pensare che un qualsiasi documento di testo i, possa essere caratterizzato da un insieme di variabili, $\boldsymbol{S}_i = \{S_{i1}, \dots, S_{ik}\}$ vettore di lunghezza K contenente tutte le word-stem di un documento i. Questo insieme $\boldsymbol{S}_i$ prende il nome di word stem profile, perciò fornisce una sintesi delle parole significative utilizzate nel documento di testo.

La quantità di interesse per l'analisi del contenuto è la percentuale di questi documenti che cadono all'interno di ogni categoria con questo termine intendiamo i sentimenti (molto negativo, negativo, neutro, ecc), P(D) = {P(D=1), …, P(D=J)}, dove P(D) è un vettore di lunghezza J, ogni elemento è una percentuale calcolata tramite tabulazione diretta:

$$\text{P(D = j)} = \quad \frac{1}{L}\sum_{l=1}^{L} \mathbf{1}\left(D_l = j\right) \quad ,$$

dove la funzione **1**(a)= $\begin{Bmatrix} 1 & se\,a\,è\,vera \\ 0 & altrimenti \end{Bmatrix}$.

I valori che la categoria D_i può assumere sono molteplici e sono scelti dall'utente, allora possiamo considera P(D) come una variabile multinomiale con J livelli possibili anche P(S) che sono tutte le possibili combinazione delle word-stem la possiamo considerare come una variabile multinomiale con 2^k livelli possibili.

La categoria del documento D_i è una variabile con molti possibili valori, scelti dall'utente, mentre il word stem profile è un insieme di variabili dicotomiche, consideriamo P(D) come una distribuzione multinomiale con J possibili valori e P(S) è una distribuzione multinomiale con 2^k possibili valori ognuno dei quali è un word stem profile.

1.2 Approcci Esistenti

Un metodo per stimare P(D) è il campionamento diretto, si sceglie la popolazione di interesse si estrae un un campione casuale e si classificano a mano tutti i documenti all'interno del campione, in fine si contano i documenti per ogni categoria. Il svantaggio principale di questo metodo è la notevole perdita di tempo.

Un secondo metodo è l'apprendimento supervisionato, l'idea alla base è di utilizzare un campione di documenti etichettato per stimare la funzione che lega la categoria, dove andrà a classificarsi il documento, e le word-stem presenti nel documento.

Perciò andiamo a prevedere la categoria, D_i variabile dipendente, con un insieme di variabili esplicative $\{S_{i1}, \ldots, S_{iK}\}$, utilizzando statistiche, support vector machine, modelli grafici o metodi di apprendimento supervisionato quali analisi discriminante lineare e quadratica, analisi basata sulla regressione logistica e modello di regressione.

Per classificare un documento quando abbiamo J categorie si utilizza la seguente formula:

$$P(\hat{D}=j)=\sum_{j'=1}^{J} P(\hat{D}=j|D=j')P(D=j')$$

dato $P(\hat{D})$ e la probabilità di errata classificazione $P(\hat{D}=j|D=j')$, questa espressione rappresenta un insieme di J equazioni, che possono essere risolte per per i J elementi di P(D).

La stima della percentuale di documenti stimati nella categoria j si calcola come, i documenti che stanno nella categoria j e effettivamente appartengo alla categoria j, perciò classificati correttamente, e documenti che stanno in altre categorie ma per una errata classificazione sono stati classificati nella categoria j.

Un problema per la discriminate e la regressione logistica accade quando il numero di predittori k è

maggiore del numero di osservazioni.

Altro modo per stimare le percentuali di classificazione è considerare una logit multinomiale o un qualsiasi altro metodo che possa generare classificazioni individuali. Stimiamo il modello per i dati dell'insieme etichettato e lo utilizziamo per classificare ogni documento della popolazione non ancora classificato. Poi si aggregano le classificazioni per ottenere una stima della percentuale di documenti in ogni categoria. Successivamente si stima la probabilità di errata classificazione da prima dividendo l'insieme dei documenti etichettati in un training set e test set. Quindi si applica lo stesso metodo di classificazione al training-set e si fanno previsioni, $\hat{D}_i$, sul test set.

Approccio da noi non utilizzato, è utilizzare il test set per stimare la probabilità di errata classificazione, tra la classificazione e la vera classe di appartenenza $P(\hat{D}_i = j | D_i = j')$. Queste probabilità, non ci dicono quali sono i documenti classificati con errore, ma possono essere usate per correggere la stima della percentuale. Esempio una volta predette le probabilità sul test-set avendo utilizzato il training-set e si riscontra che il 17% dei documenti sono stati classificati come D=1 quando realmente dovrebbero essere classificati come D=2. Per correggere la stima basta sottrarre questa quantità dalla categoria 1 e aggiungerla alla categoria 2.

1.3 Problemi

Purtroppo l'utilizzo di questi metodi per stimare le aggregazioni di documenti provocano dei problemi. Come sottolinea Hand (2006), l'apprendimento supervisionato per la classificazione di singoli documenti fallisce in alcuni casi, entrambi comuni nella pratica:

- Quando il campione etichettato non è un campione casuale semplice dalla popolazione i metodi falliscono. Deviazioni dalla casualità possono verificarsi a causa della “deriva della popolazione”, cioè quando l'insieme etichettato viene raccolto in un determinato periodo di tempo e lo si vuole applicare alla popolazione raccolta nel tempo, andando incontro a delle problematiche come la sintassi che cambia evolvendosi.
- La funzione prevista dall'apprendimento supervisionato prevede D dato S, cioè modellare P(D | S), ma in realtà si deve ragionare al contrario cioè P(S | D) cioè prevedere S data la categoria D. La conseguenza dell'utilizzo di P(D | S) è basato sul requisito di due ipotesi necessarie per generalizzare dal campione etichettato alla popolazione.

  La prima, S attraversa lo spazio di tutti i predittori di D, significa che una volta controllate le variabili non esistono altre variabili che potrebbero migliorare il potere predittivo.

  Altra ipotesi che la classe del modello scelta per P(D | S) includa il vero modello.

  Ma trovare un buon modello o quello esatto non è per nulla facile e molto dispendioso in termini di tempo.

CAPITOLO II

2.1 Metodo Gary King e Hopkins Daniel.J: ReadMe

Il metodo creato da Daniel J.Hopkins e Gary King, che chiameremo ReadMe dato che il software si chiama così, tratta ***S*** come conseguenza di *D* evitando di dover calcolare la stima $\hat{D}$ ma usando ***S*** al suo posto, visto che $\hat{D}$ è funzione di **S**. Allora troviamo:

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

scrivendola in forma di matrice risulta:

$$\underset{2^K x\,1}{P(\boldsymbol{S}=s)}=\underset{2^K x\,J}{(\boldsymbol{S}|D)}\underset{Jx1}{P(D)} \qquad (2)$$

dove *P(**S**)* indica la probabilità di ognuna delle 2^K possibili word-stem pofiles (***S***). Per esempio se *K*=3 word stem, *P(**S**)* contiene le probabilità di ognuna delle $2^3 = 8$ combinazioni possibili, da quella che non contiene nessuna word-stem a quella che le contiene tutte, quindi tutte le combinazioni possibili sono, (000, 001, 010, 011, 100, 101, 110, 111).

P(S | D) è la probabilità di ognuna delle 2^K possibili word-stem profiles sia all'interno della categoria *D*, mentre *P(D)* è il nostro vettore di interesse.

2.2 Sintesi del processo di stima

In questo paragrafo faremo una sintesi del processo di stima e nel paragrafo successivo lo vedremo in dettaglio con un esempio.

Gli elementi di *P(**S**)* possono essere stimati sulla base delle frequenze osservate sulla popolazione, ci limiteremo a calcolare la percentuale di documenti osservati con ciascun word-stem profiles. Dato che *D* non è osservabile dalla popolazione non siamo in grado di stimare *P(**S** | D)* allora si assume che questa quantità è uguale anche nel campione etichettato, cosi da poter stimare la matrice:

$$P^h(\boldsymbol{S} \mid D) = P(\boldsymbol{S} \mid D) \qquad (3)$$

Potremmo stimare *P(D)* dall'equazione (2), assumendo vera l'equazione (3) e l'accuratezza delle stime di *P(S)* e di *P(S|D)*, risolvendo l'equazione (2) tramite regressione lineare.

Possiamo pensare che *P(D),* che non conosciamo, siano i coefficienti di regressione *β*, *P(S|D)* come variabili esplicative matrice *X* e *P(S)* come variabili dipendenti Y, allora l'equazione [1] diventa $Y=X\beta$. Per stimare i coefficienti di regressione si utilizza l'usuale formula $\beta= (X^TX)^{-1}X^Ty$. Il vantaggio sta nel fatto che non occorre classificare i singoli documenti dentro le categorie ma esso stima direttamente le percentuali.

Questo semplice approccio ha delle difficoltà nell'essere applicato:

- K è normalmente un numero grande allora 2^K sarà un numero enorme, impossibile da gestire anche per i computer. Per esempio K=30, cioè pochissime parole, allora 2^K=1.073.741.824.
- il numero di osservazioni disponibili per stimare P(S) e P(S|D) è molto più piccolo di del potenziale numero di word-stem profile, $n<<2^K$.

Per evitare questi problemi si sceglie casualmente un sottoinsieme di parole tra le 5 e le 25 in modo da ridurre il carico computazionale.

Una volta determinato il numero ottimale di parole per sottoinsieme attraverso cross validazione, risolviamo per ogni P(D) e si fa la media dei risultati ottenuti per sottoinsieme.

2.3 Esempio

Per capire meglio come funziona la prima parte del metodo vediamo un esempio.

Da una popolazione di interesse, tutti i tweets scaricati, si estrae un piccolo campione che chiamiamo Training Set, nel esempio i 5 tweets che seguono, nella reale applicazione del metodo i tweets possono essere svariate centinaia.

Questi riguardano la politica in Italia, nello specifico andremo a stimare il gradimento di Monti:

[1] "Il #berlusconismo ha indotto un'analfabetismo politico tale che, di fronte a #Monti, siamo tutti casalinghe di Voghera. #politica".

[2] "Sulla #costa crociere sesso, droga e machismo da parte degli ufficiali... Ma non è che #Berlusconi è tornato a lavorare sulle navi? ".

[3] "@rosy_bindi @pierferdinando bravi continuate con questa logica del secolo scorso, continuate a rovinare il paese".

[4] "J Ax querela ancora #silvio, il nuovo inno del #pdl è uguale a \"era meglio prima\" ".

[5] "Prof.#Monti, lei che è esperto d'iniquità, scusi, di equità fiscale, spieghi: no IMU su attività no profit Chiesa. Che profit ho dalla casa?"

Per poter applicare il metodo ReadMe a questi tweets bisogna considerare che inizialmente essi sono "sporchi" e che pertanto vanno "puliti", per la corretta applicazione e interpretazione della formula. La fase di pulizia, fatta da algoritmi consiste nell'eliminare tutto ciò che è superfluo ai fini della analisi, questa comprende varie fasi:

- La trasformazione di tutte le lettere maiuscole in minuscole,

- L'eliminazione dal documento delle parole vuote “stopwords” che sono necessarie per la corretta costruzione della frase ma sono vuote di significato. Appartengono a questa classe gli articoli, le congiunzioni, le preposizioni e le interiezioni.
- Il riconoscimento di polimorfismi, sequenza di parole che esprime un concetto autonomo, differente da quello espresso dalle singole parole.

Le fasi appena descritte sono relativamente semplici da realizzare/applicare, a questo sarebbe da aggiungere lo stemming ma risulta molto più complicato eseguirlo per la lingua italiana, cosa che invece risulta “facilmente” applicabile nella lingua inglese in quanto lo stemming è il processo di riduzione della forma flessa di una parola alla sua forma radice, detta tema. Il tema non corrisponde necessariamente alla radice morfologica della parola, normalmente è difficile che le parole correlate siano mappate allo stesso tema (ad esempio: andare, andai, andò vengono mappate con il tema and, anche se and non è una valida radice per la parola)

Altra cosa difficile da eseguire è la fase di specificazione degli anagrammi. Come ad esempio “campagna” e “elettorale” singolarmente hanno molti significati, ma se sono entrambi presenti all'interno di una frase e in successione “campagna elettorale” ecco che ha un significato ben preciso, quindi ai fine della analisi le potremmo unire formando un'unica parola “campagnaelettorale”.

[1] "#berlusconismo indotto analfabetismo politico tale fronte #monti casalinghe voghera #politica".

[2]”#costa crociere sesso droga machismo parte ufficiali silvioberlusconi tornato lavorare navi”.

[3] “rosybindi pierferdinandocasini bravi continuate logica secolo scorso continuate rovinare paese”.

[4] “ax querela #silvio inno popolodellelibertà uguale prima”.

[5] "prof #monti esperto iniquità scusi equità fiscale spieghi imu attività profit chiesa profit casa".

Una volta puliti i twetts si classificano secondo il gradimento rispetto a Monti, i valori possibili in questo esempio sono definiti nella seguente tabella :

CATEGORIE	***GRADIMENTO***
1	Positivo
0	Neutrale
-1	Negativo
NA	Non attinente ai fini

Come si può vedere le categorie sono 4, (1, 0, -1, NA), perciò J=4. I cinque tweets possono essere classificati come segue:

N° TWEET	CLASSIFICAZIONE
1	0
2	NA
3	NA
4	NA
5	-1

Ora ogni singolo tweet lo possiamo vedere come un'insieme di parole tutte contenute nel word-stem profile associato $\mathbf{S}_i$. Per fare questo prima si crea un insieme con tutte le parole presenti nel sottoinsieme, come si vede nella seguente tabella:

	#berlusconismo	#monti	#politica	analfabetismo	attività	ax	bravi	casa	casalinghe	chiesa	continuate	costacrociere	droga	equità	esperto	fiscale	fronte	imu	indotto	iniquità	inno	lavorare	logica	machismo	navi	paese	parte	pierferdinandocasini	politico	popolodellelibertà	prima	profit	querela	rosybindi	rovinare	scorso	scusi	secolo	sesso	silvioberlusconi	spieghi	tale	tornato	ufficiali	uguale	vogliera
S1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
S2	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0
S3	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0
S4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0
S5	0	1	0	0	1	0	0	1	0	1	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0

come possiamo vedere $\mathbf{S}_i$ è un vettore contenente 0 e 1 in base alla presenza o meno della parola all'interno del tweet, da notare che con questo metodo si indica solo la presenza della parola non la frequenza con cui questa si ripete all'interno del tweet.

Nella seguente tabella mettiamo in evidenza gli word-stem profile e le categorie ad esse associate:

WORD STEM PROFILE	*CLASSIFICAZIONE*
S1	0
S2	NA
S3	NA
S4	NA
S5	-1

Per facilitare l'esposizione dei prossimi passaggi, da tutte le word-setm ne estraiamo alcune. Le estraiamo in modo che riescano a definire in modo univoco la categoria di appartenenza cosi da non avere informazioni ridondanti come si vede nella seguente tabella:

WORD STEM PROFILE	***WORD STEM***				
	analfabetismo	***bravi***	***fiscale***	***lavorare***	***profit***
S1	1	0	0	0	0
S2	0	0	0	1	0
S3	0	1	0	0	0
S4	0	0	0	0	0
S5	0	0	1	0	1

Ora si creano tutte le possibili combinazioni di word-stem, che in questo caso saranno $2^5 = 32$, perciò avremmo 32 possibili word-stem profile come si vede nella seguente tabella:

S	***analfabetismo***	***bravi***	***fiscale***	***lavorare***	***profit***
00000	0	0	0	0	0
00001	0	0	0	0	1
. . . . .	.	.	.	.	.
. . . . .	.	.	.	.	.
11110	1	1	1	1	0
11111	1	1	1	1	1

Quindi il vettore ***S*** di lunghezza 32 conterrà tutte le possibili combinazioni.
Considerando la seguente formula vista anche in precedenza:

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

ed applicandola all'esempio dove $J = 4$, otteniamo:

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

scritta in forma estesa otteniamo:

$$P(\boldsymbol{S}=j)= P(\boldsymbol{S}=s|D=1)P(D=1)+P(\boldsymbol{S}=s|D=0)P(D=0)+ \\ + P(\boldsymbol{S}=s|D=-1)P(D=-1)+P(\boldsymbol{S}=s|D=NA)P(D=NA)$$

le quantità di interesse sono $P(D=1), P(D=0), P(D=-1), P(D=NA)$.
Data una combinazione di word-stem profile ***S***, proveniente dalla popolazione ancora da classificare, il metodo restituisce le percentuali di appartenenza ad ogni singola categorie, in questo esempio sono -1, 0, 1 e NA.

2.4 Problematiche di ReadMe e generali delle Sentiment Analysis

Le principali problematiche dell'applicazione di questo metodo è la specificazione dei bigrammi e trigrammi. Bigrammi e trigrammi sono anagrammi definiti per ridurre il numero delle word-stem totali K.
Tutt'altro genere di problematiche sono i casi dove il metodo non funziona correttamente:

- ambiguità lessicale, esempio “Flavio Roma fantastico” sebbene questo tweet sia valutato correttamente in modo positivo, il metodo non riconosce la differenza tra Roma (città) e Flavio Roma (portiere).
- mancato riconoscimento della punteggiatura, “paperino antipatico?” in questo caso non compare nessuna forma di gradimento ma il metodo lo legge “paperino antipatico” che chiaramente sarà interpretato con un gradimento negativo nei confronti di paperino.
- incapacità di riconoscere le espressioni dal significato “non letterale” come le frasi il cui tono è sarcastico e ironico.

CAPITOLO III

3.1 Applicazione ai dati reali

L'elaborazione, come accennato nell'introduzione, riguarda i tweets pubblicati durante l'evento televisivo, “Primarie centrosinistra”, andato in onda il giorno 11 novembre 2012 su SKY.

Dalla l'intera popolazione dei tweets sono stati presi in considerazione solo tweets riferiti a Pier Luigi Bersani o Matteo Renzi, questa sarà la nostra popolazione di riferimento. Da notare che si sono esclusi dalla popolazione di riferimento i tweets riferiti ad entrambi i candidati. Da questa popolazione di riferimento sono stati estratti due campioni di 1000 tweets ciascuno per ogni candidato. Ogni campione è stato interamente classificato e suddiviso in training set (800 tweets) e test set (200 tweets). Infine è stato creato un data set per ogni campione.

In seguito ho analizzato questi dataset più volte utilizzando strategie e analisi differenti per confrontarle e trovare quella che al meglio rappresentasse le informazioni raccolte.

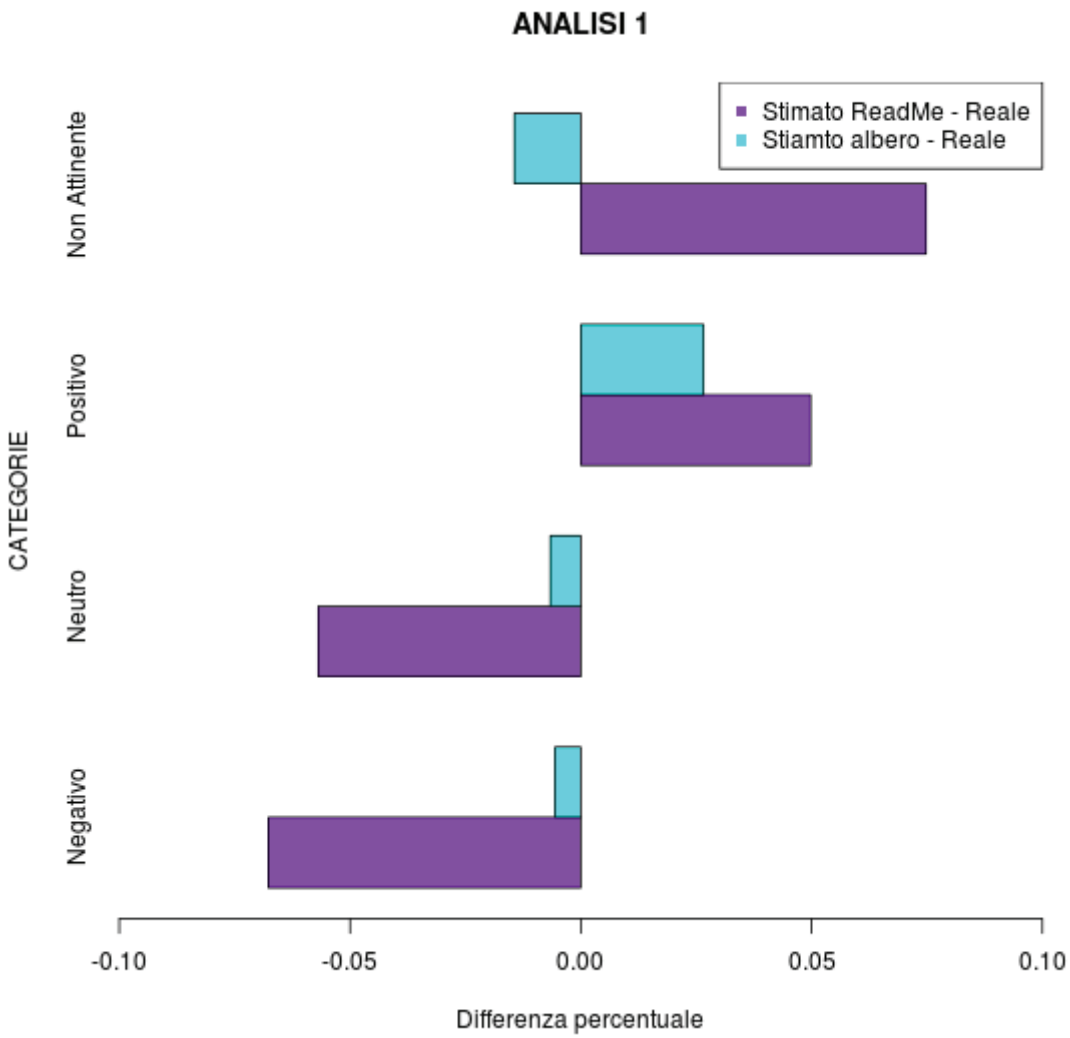
Analisi 1: è stata fatta utilizzando un unico dataset di 2000 tweets equamente divisi tra Pier Luigi Bersani e Matteo Renzi. Le classificazione in questa analisi non tengono conto del soggetto, quindi avremo solamente quattro categorie (Negativo, Neutro, Positivo e Non attinente) e un tweets classificato può appartenere a solo una di esse.

A questo data è stato applicato il metodo ReadMe e poi l'albero di classificazione.

I risultati ottenuti dai due differenti metodi sono riportati nella seguente tabella:

	Negativo	*Neutro*	*Positivo*	*Non attinente*
% Reale	**0.2525**	**0.2725**	**0.2050**	**0.2700**
% Stimata ReadMe	0.1848	0.2156	0.2548	0.3448
% albero	0.2469	0.2659	0.2316	0.2556

Questo grafico rappresenta la differenza tra percentuale stimata e percentuale reale per entrambi i metodi:



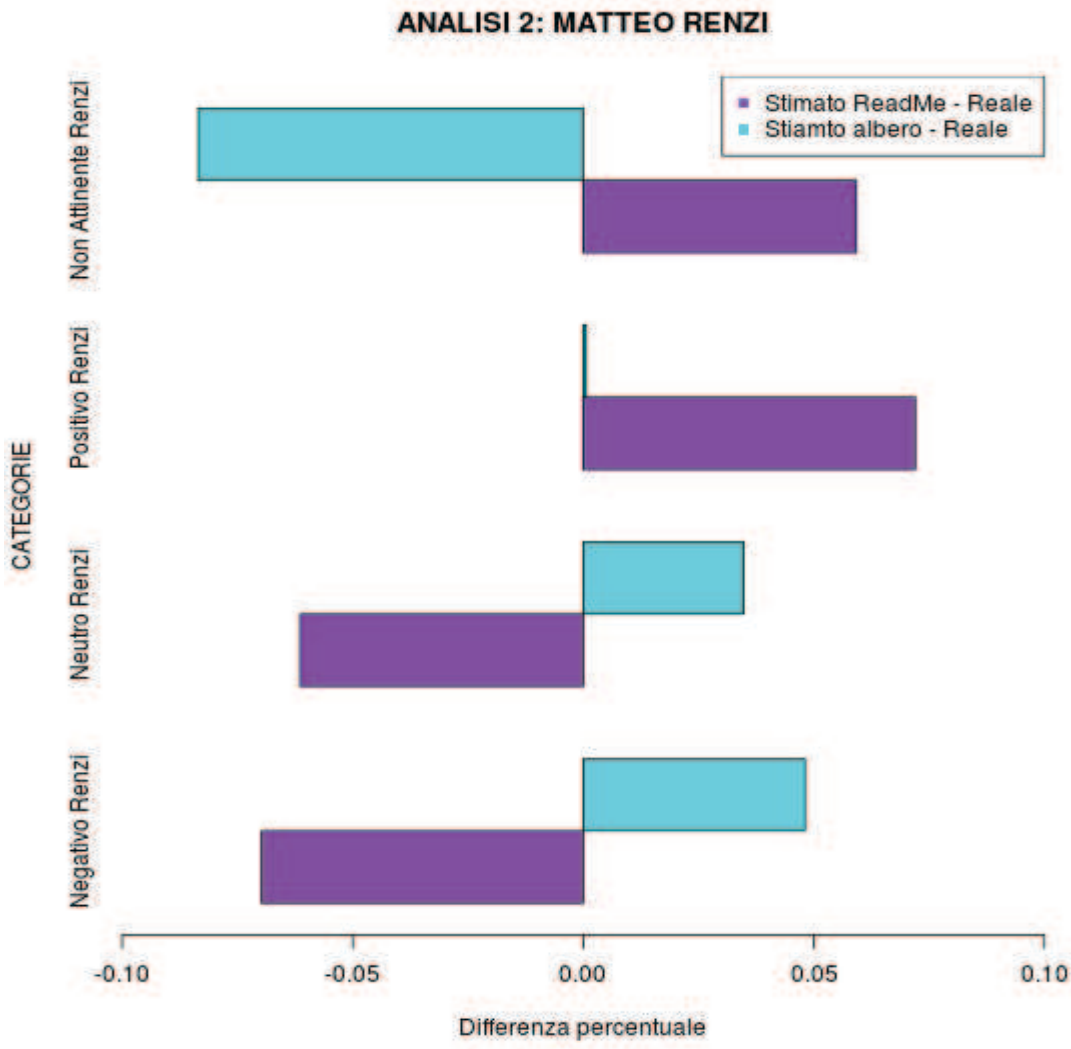
Come indice di qualità delle stime possiamo utilizzare la somma degli scarti al quadrato, tra stimati è reali. Ottenendo tra stimati di ReadMe e reali 0.0159, mentre per stimati dell'albero e reali 0.0010, notiamo che l'albero di classificazione è più efficiente di ReadMe.

Analisi 2: è stata fatta applicando il metodo ReadMe ad ognuno dei dataset e poi confrontando i risultati con l'albero di classificazione. Ogni dataset in questa analisi ha categorie differenti, per il dataset riferito a Matteo Renzi esse sono (Negativo Renzi, neutro Renzi, positivo Renzi e non attinente Renzi), mentre per il dataset riferito a Pier Luigi Bersani le categorie sono (Negativo Bersani, Neutro Bersani, Positivo Renzi o Non attinente Bersani).
Al dataset riguardante solo Matteo Renzi è stato applicato il metodo ReadMe e poi l'albero di classificazione.

I risultati ottenuti dai due differenti metodi sono riportati nella seguente tabella:

	Negativo Renzi	*Neutro Renzi*	*Positivo Renzi*	*Non attinente Renzi*
% Reale	**0.215**	**0.220**	**0.225**	**0.340**
% Stimata ReadMe	0.1452	0.1586	0.2970	0.3992
% albero	0.2631	0.2548	0.2256	0.2565

Questo grafico rappresenta la differenza tra percentuale stimata e percentuale reale per entrambi i metodi:



Come metodo di paragone possiamo utilizzare la somma degli scarti al quadrato, tra stimati è reali. Ottenendo tra stimati di ReadMe e reali 0.0158, mentre per stimati dell'albero e reali 0.0105,

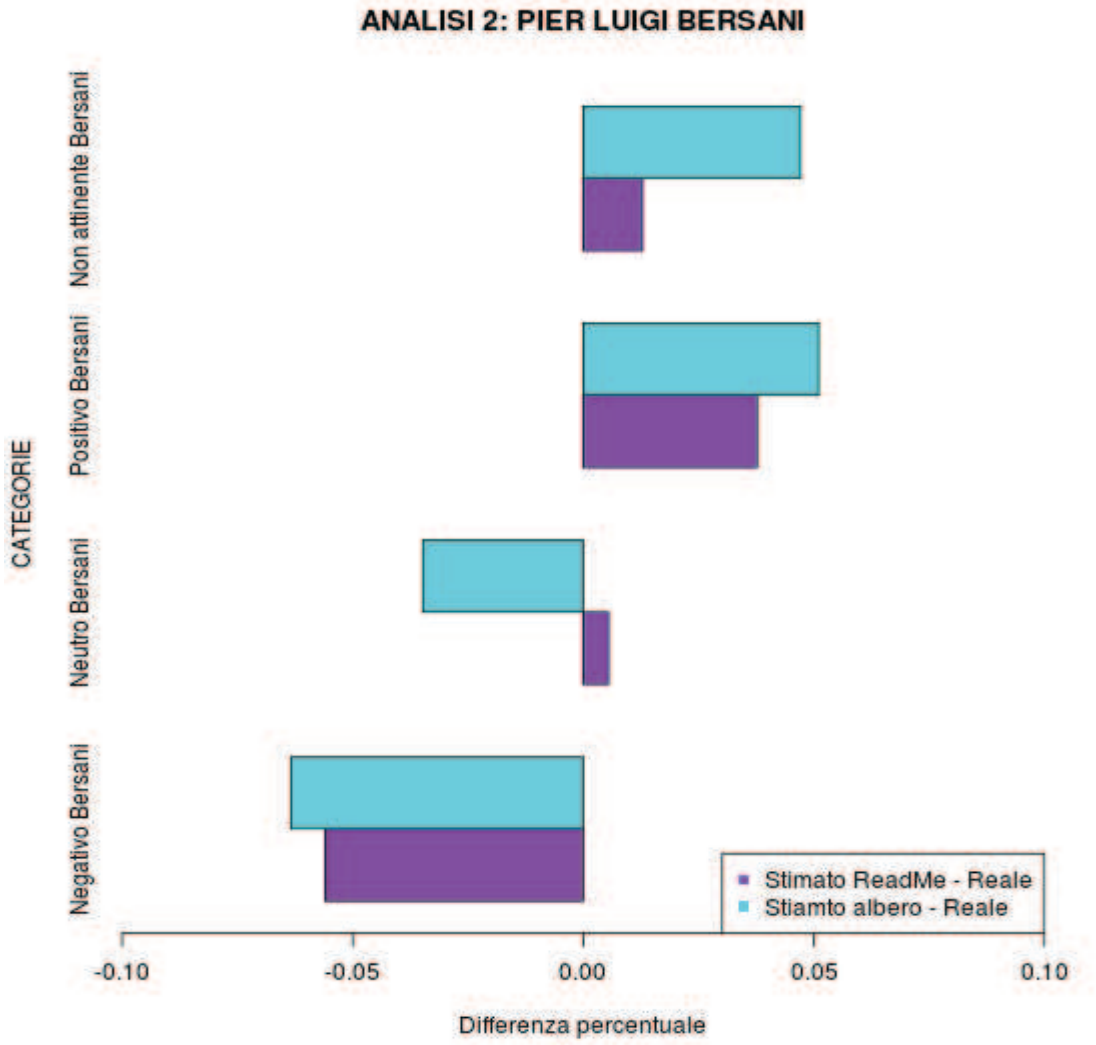
l'albero di classificazione è più efficiente seppure di poco rispetto a ReadMe, possiamo dire che all'incirca si equivalgono dato che sono dello stesso ordine di grandezza.

Al dataset riguardante solo Pier Luigi Bersani è stato applicato il metodo ReadMe e poi l'albero di classificazione.

I risultati ottenuti dai due differenti metodi sono riportati nella seguente tabella:

	Negativo Bersani	*Neutro Bersani*	*Positivo Bersani*	*Non attinente Bersani*
% Reale	**0.290**	**0.325**	**0.185**	**0.200**
% Stimata ReadMe	0.2340	0.3305	0.2227	0.2128
% Albero	0. 2266	0. 2902	0. 2361	0. 2471

Questo grafico rappresenta la differenza tra percentuale stimata e percentuale reale per entrambi i metodi:



Come metodo di paragone possiamo utilizzare la somma degli scarti al quadrato, tra stimati è reali. Ottenendo tra stimati di ReadMe e reali 0.0047, mentre per stimati dell'albero e reali 0.0101, in questo caso ReadME è più efficiente rispetto dell'albero.

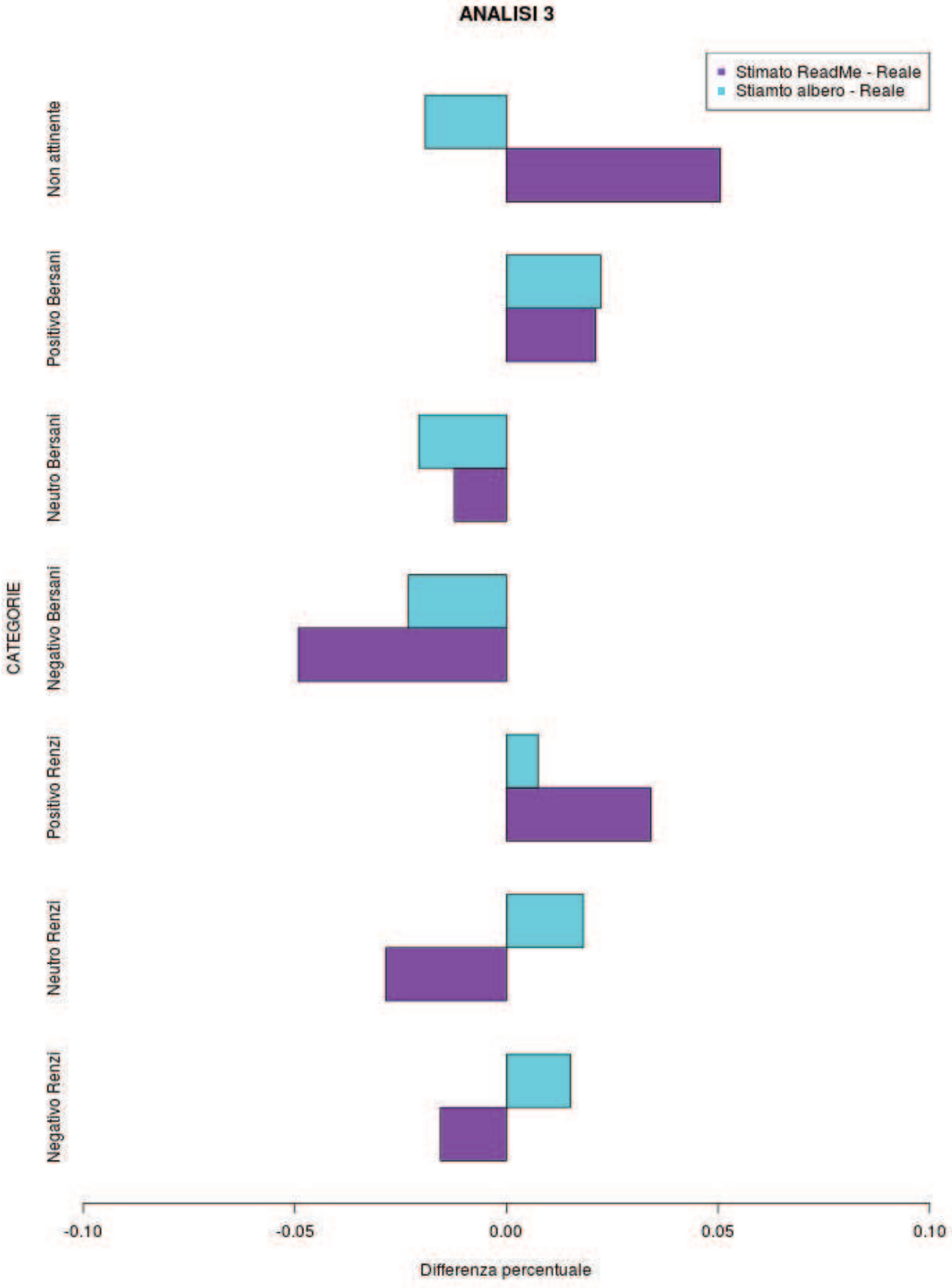
Analisi 3: è stata fatta unendo i due dataset, tenendo le categorie della analisi 2, ottenendo così 7 categorie, per la parte riferita a Matteo Renzi (Negativo Renzi, neutro Renzi, positivo Renzi e non attinente Renzi), mentre per la parte riferita a Pier Luigi Bersani le categorie sono (Negativo Bersani, Neutro Bersani, Positivo Renzi o Non attinente Bersani), le categorie Non attinente Renzi e Non attinente Bersani vengono fuse ottenendo una solo categoria Non attinente.
Su questo nuovo dataset è stato applicato il metodo ReadMe e poi l'albero di classificazione.

I risultati ottenuti dai due differenti metodi sono riportati nella seguente tabella:

	Renzi			***Bersani***			
	Negativo	*Neutro*	*Positivo*	*Negativo*	*Neutro*	*Positivo*	*Non attinente*
% Reale	**0.1075**	**0.1100**	**0.1125**	**0.1450**	**0.1625**	**0.0925**	**0.2700**
% Stimata ReadMe	0.0919	0.0815	0.1466	0.0958	0.1502	0.1135	0.3205
% Albero	0.1226	0.1282	0.1200	0.1218	0.1418	0.1148	0.2508

Come metodo di paragone possiamo utilizzare la somma degli scarti al quadrato, tra stimati è reali. Ottenendo tra stimati di ReadMe e reali 0.0078, mentre per stimati dell'albero e reali 0.0024, in questo caso l'albero è più efficiente seppure di poco rispetto a ReadMe, ma possiamo dire che all'incirca si equivalgono dato che sono dello stesso ordine di grandezza.

Questo grafico rappresenta la differenza tra percentuale stimata e percentuale reale per entrambi i metodi:



CAPITOLO IV

4.1 Conclusioni

L'obbiettivo del nostro studio era valutare le prestazioni di ReadMe rispetto all'albero di classificazione.

Riportiamo nella seguente tabella le somme degli scarti al quadrato ottenuti per ogni analisi. Per poter confrontare meglio le tre analisi occorre sommare i risultati parziali per Matteo Renzi e Pier Luigi Bersani riferiti all'analisi 2.

<table>
<tr><th rowspan="2"></th><th rowspan="2">Analisi 1</th><th colspan="2">Analisi 2</th><th rowspan="2">Analisi 3</th></tr>
<tr><th>Renzi</th><th>Bersani</th></tr>
<tr><td rowspan="2">Stimati ReadMe - Reali</td><td rowspan="2">0.0159</td><td>0.0158</td><td>0.0047</td><td rowspan="2">0.0078</td></tr>
<tr><td colspan="2">0.0204</td></tr>
<tr><td rowspan="2">Stimati albero - Reali</td><td rowspan="2">0.0010</td><td>0.0105</td><td>0.0101</td><td rowspan="2">0.0024</td></tr>
<tr><td colspan="2">0.0206</td></tr>
</table>

Analizziamo la tabella per colonne, quindi per analisi:

- Per l'analisi 1, vediamo che l'albero di classificazione è più efficace nel stimare le percentuali rispetto a ReadMe.
- Per l'analisi 2, utilizzare ReadMe o l'albero di classificazione è pressoché equivalente.
- Per l'analisi 3, si ha una leggero vantaggio nell'usare l'albero di classificazione ma anche qui utilizzare l'uno o l'altro è pressoché equivalente.

Quello che possiamo concludere basandosi sulle analisi fatte da questa tabella è di non basarsi solamente su un metodo per stimare le percentuali, ma di utilizzarli entrambi ed eseguirli più volte, facendo poi una media sulle percentuali ottenute per ogni categoria, almeno per quanto riguarda ReadMe.

Fino ad ora abbiamo analizzato i campioni estratti dalla popolazione di riferimento e non abbiamo detto nulla sulle stime di essa. Prendendo come riferimento la strategia utilizzata nell'analisi 3, visto che sembra essere la migliore, la utilizzeremo per stimare le percentuali all'interno della popolazione di riferimento, circa 8400 tweets, riguardati solo Matteo Renzi o Pier Luigi Bersani.

I risultati ottenuti sia con il metodo ReadMe e dell'albero sono rappresentati nella seguente tabella

	Renzi			***Bersani***			
	Negativo	*Neutro*	*Positivo*	*Negativo*	*Neutro*	*Positivo*	*Non attinente*
% Stimata ReadMe	0.1381	0.1317	0.1556	0.087	0.0957	0.0698	0.3218
% Albero	0.1554	0.1653	0.1458	0.0932	0.1051	0.1457	0.2508

Alcuni aspetti richiederebbero ulteriori approfondimenti e ricerche, al fine di migliorare l'approccio al metodo. Ad esempio si potrebbero considerare analisi che non considerano i RT (ritwittato), nel nostro caso rappresentano il 47.5% dei 2000 tweets, utilizzati come campione, mentre rappresentano circa il 48% dell'intera popolazione di riferimento.

APPENDICE A Funzioni 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 è:

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.

INPUT	
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	É un numero compreso tra 0 ed 1 ed indica una percentuale, verrano incluse solo le parole che superano questa soglia. Per includere tutte le parole settiamo threshold a 0. Di default è 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 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 e FALSE.
sep	Indica il separatore che si utilizza per separare le colonne in “*control.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.

In questa tabella si vedono gli output della funzione.

OUTPUT	
trainingset	tabella binaria che indica quali parole, tra quelle che soddisfano la clausola threshold, appaiono in ogni testo del training-set.
testset	Stesso formato del training-set, ma per il test-set.
formula	Formula da utilizzare nella chiamata alla funzione VA. Scritta secondo lo standard di 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 dipendenti. “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 a 15.
n.subset	Numero di sottoinsiemi 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 uguale ad 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

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

La sintassi è:

preprocess(undergrad.results)

In questa tabella si vedono gli input.

INPUT	
undergrad.results	L'output da readme.

In questa tabella si vedono gli output.

OUTPUT	
undergrad.preprocessed	Una lista pre elaborata con colonne costanti rimosse.

A.3 Funzione 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(), trainingse = NULL, testset = NULL, formula = NULL, n.subset = NULL, prob.wt = NULL, boot.se = NULL, nboot = NULL, printit = NULL)

In questa tabella vengono riportati i parametri di ingresso alla funzione.

INPUT	
undergradlist	Una lista che ha un elemento per ogni parametro. Specificare i valori dei parametri o attraverso una lista o attraverso 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 impostato 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	Numero positivo 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 deve avere se si seleziona il sottoinsieme di caratteristiche. Quando prob.wt è un vettore, 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 bootstrap standard errors di CSMF sono stimati. Questa opzione richiede generalmente molto tempo per l'esecuzione. Di default è impostata a FALSE.
nboot	Numero intero positivo. Se boot.se=TRUE, esso specifica 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

In questa tabella vengono riportati gli output della funzione.

OUTPUT	
est.CSMF	La stima della percentuale in 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 creazione ed analisi per il candidato Matteo Renzi*.

```
rm(list=ls())
# Carichiamo i dati, inserendo il percorso dove sono salvati i dati.
load("... /R/x86_64-pc-linux-gnu-library/2.14/datiFabio.Rdata")
# Cerchiamo l'id dei tweets dove compare solo la parla bersani.
idBersani=grep("bersani", db$text)
# Cerchiamo l'id dei tweets dove compare solo la parla renzi.
idRenzi= grep("renzi", db$text)
# unione dei due insiemi idBersani e idRenzi
unioneBersaniRenzi= union(idBersani, idRenzi)
# intersezione tra i due insiemi
intersezioneBersaniRenzi= intersect(idBersani,idRenzi)
# differenza tra unione ed intersezione, dall'unione tolgo gli elementi dell'intersezione, in modo da
# avere solo tweets che contengono o bersani o renzi.
idBersaniRenzi= setdiff(unioneBersaniRenzi,intersezioneBersaniRenzi)
# creiamo il database con solo i dati relativi a tweet precedentemente cercati
dbBersaniRenzi=db[idBersaniRenzi,]
head(dbBersaniRenzi)
idSoloRenzi = idBersaniRenzi[2354:6405]
head(idSoloRenzi)
dbSoloRenzi = db[idSoloRenzi,]
head(dbSoloRenzi)
# scegliamo quanti tweets prendere tra tutti i disponibili dal dataset riferito a renzi.
x=1000
# creiamo il campione.
sampleRenzi= sample(nrow(dbSoloRenzi), x)
#vedo quali sono gli id dei tweets.
dbSoloRenzi$orig[sampleRenzi]
# ATTENZIONE CAMBIARE LA WD. Creata cartella PostRenzi
Setwd("... /R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/PostRenzi")
# Creo tutti i singoli file di testo.
sapply(sampleRenzi,function(id) cat(dbSoloRenzi$text[id],file=paste(id,sep="",".txt")))
# Creo un file di testo che contine tutti i file di testo creati prima e gli classifico a mano.
write.csv(file="controlPre.txt",cbind(FILENAME=paste(sampleRenzi,".txt",sep=""),TRUTH=NA,
```

```
+ TRAININGSET=rep(1:0,c(800,200)), message=dbSoloRenzi$orig[sampleRenzi]),
+ row.names=FALSE)
################################ ##################
# CASSIFICARE I SINGOLI FILE DI TESTO NEL FILE CONTROLPRE.TXT  <<---     #
##################################################
# leggo il file di testo controlPre.txt, appena classificato.
testo=read.csv(... / R/ x86_64-pc-linux-gnu-library/ 2.14/ ReadMe/ PrimarieDiSinistra/ PostRenzi/
+controPre.csv)
str(testo)
attach(testo)
#creo un data frame con solo le prime tre colonne del file controlPre.csv.
control = data.frame(FILENAME,TRUTH,TRAININGSET)
write.table(control, file="control.txt",row.names=FALSE, sep=",", quote=FALSE)
# ora creo il file tableFile.txt.
library(ReadMe)
library(quadprog)
oldwd <- getwd()
oldwd
# impostare la WD .../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/ PrimarieDiSinistra/ postRenzi
setwd("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/PostRenzi")
getwd()
undergrad.results <- undergrad(sep = ",")
undergrad.results
undergrad.preprocess <- preprocess(undergrad.results)
undergrad.preprocess
# vediamo quante colonne sono state rimosse
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),
+ length(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))
readme.results <- readme(undergrad.preprocess)
readme.results
#stima della percentuale in ogni categoria.
readme.results$est.CSMF
# percentuale reale.
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)
par(mar=c(1,1,1,1), xpd=TRUE)
plot(albero)
text(albero,all=TRUE,use.n=TRUE)
# Stima dell'errore
plotcp(albero)
printcp(albero)
# Stima nel TestSet
new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])
head(new.stem)
potatura=prune(albero, cp=0.09)
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 = sum((predetti-reali)^2)
scartiReadMeAlbero=sum((media - reali)^2)
```

B.2 *Script per creazione ed analisi per il candidato Pier Luigi Bersani*.

I procedimenti sono uguali a quelli fatti per Matteo Renzi.

```
rm(list=ls())
# Carichiamo i dati, inserendo il percorso dove sono salvati i dati.
load("... /R/x86_64-pc-linux-gnu-library/2.14/datiFabio.Rdata")
# Cerchiamo l'id dei tweets dove compare solo la parla bersani.
idBersani=grep("bersani", db$text)
# Cerchiamo l'id dei tweets dove compare solo la parla renzi.
idRenzi= grep("renzi", db$text)
# unione dei due insiemi idBersani e idRenzi
unioneBersaniRenzi= union(idBersani, idRenzi)
# intersezione tra i due insiemi
intersezioneBersaniRenzi= intersect(idBersani,idRenzi)
# differenza tra unione ed intersezione, dall'unione tolgo gli elementi dell'intersezione, in modo da
# avere solo tweets che contengono o bersani o renzi.
idBersaniRenzi= setdiff(unioneBersaniRenzi,intersezioneBersaniRenzi)
# creiamo il database con solo i dati relativi a tweet precedentemente cercati
dbBersaniRenzi=db[idBersaniRenzi,]
head(dbBersaniRenzi)
idSoloBersani = idBersaniRenzi[1:2353]
length(idSoloBersani)
head(idSoloBersani)
dbSoloBersani = db[idSoloBersani,]
head(dbSoloBersani)
# scegliamo quanti tweets prendere tra tutti i disponibili dal dataset riferito a bersani.
y=1000
# creiamo il campione.
sampleBersani= sample(nrow(dbSoloBersani), y)
#vedo quali sono gli id dei tweets.
dbSoloBersani$orig[sampleBersani]
# ATTENZIONE CAMBIARE LA WD. Creata cartella PostBersani.
setwd("... /R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/PostBersani")
# Creo tutti i singoli file di testo
sapply(sampleBersani,function(id) cat(dbSoloBersani$text[id],file=paste(,id,sep="",".txtB")))
# Creo un file di testo che contiene tutti i file di testo creati prima.
```

```
write.csv(file="controlPre.txt",cbind(FILENAME=paste(sampleBersani,".txtB",sep=""),
        +TRUTH=NA,TRAININGSET=rep(1:0,c(800,200)),
        +message=dbSoloBersani$orig[sampleBersani]), row.names=FALSE)
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# CASSIFICARE I SINGOLI FILE DI TESTO NEL FILE CONTROLPRE.TXT  <<---   #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# leggo il file di testo controlPre.txt.
testo = read.csv(file.choose())
str(testo)
attach(testo)
#creo un data frame con solo le prime tre colonne del file controlPre.txt.
control = data.frame(FILENAME,TRUTH,TRAININGSET)
write.table(control, file="control.txt",row.names=FALSE, sep=",", quote=FALSE)
# ora creo il file : tableFile.txt
library(ReadMe)
library(quadprog)
oldwd <- getwd()
oldwd
# impostare la WD … /R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/
 # PostBersani.
setwd("... /R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/PostBersani")
getwd()
undergrad.results <- undergrad(sep = ",")
undergrad.results
undergrad.preprocess <- preprocess(undergrad.results)
undergrad.preprocess
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),
        + length(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))
readme.results <- readme(undergrad.preprocess)
readme.results
#stima della percentuale in ogni categoria.
readme.results$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

l*ibrary(rpart)*

albero = rpart(verita~., data=x, method="class")

summary(albero)

albero

print(albero)

par(mar=c(1,1,1,1), xpd=TRUE)

plot(albero)

text(albero,all=TRUE,use.n=TRUE)

Stima dell'errore

plotcp(albero)

printcp(albero)

Stima nel TestSet

new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])

head(new.stem)

potatura=prune(albero, cp=0.09)

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 = sum((predetti-reali)^2)

scartiReadMeAlbero=sum((media - reali)^2)

B.3 *Script Renzi e Bersani 7 categorie.*

Una buona parte del codice è già spiegata in A.4.2 e in A.4.1.

```
rm(list=ls())
# percorso dove sono salvati i dati.
load("... /R/x86_64-pc-linux-gnu-library/2.14/datiFabio.Rdata")
ls()
idBersani=grep("bersani", db$text)
length(idBersani)
idRenzi= grep("renzi", db$text)
length(idRenzi)
# unione dei due insiemi idBersani e idRenzi
unioneBersaniRenzi= union(idBersani, idRenzi)
# intersezione tra i due insiemi
intersezioneBersaniRenzi= intersect(idBersani,idRenzi)
# differenza tra unione ed intersezione, dall'unione tolgo gli elementi dell'intersezione
idBersaniRenzi= setdiff(unioneBersaniRenzi,intersezioneBersaniRenzi)
dbBersaniRenzi=db[idBersaniRenzi,]
head(dbBersaniRenzi)
# prendo solo gli id riferiti Bersani.
idSoloBersani=idBersaniRenzi[1:2353]
dbSoloBersani = db[idSoloBersani, ]
# prendo gli id riferiti a Renzi.
idSoloRenzi = idBersaniRenzi[2354:6405]
dbSoloRenzi = db[idSoloRenzi,]
# carico i file controlPre.csv si Renzi e Bersani.
controlPreRenzi=read.csv(".../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/
        +PostRenzi/controlPre.csv")
controlPreBersani=read.csv(".../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/
        +PostBersani/controlPre.csv")
# unisco per righe i due file.
control = rbind(controlPreRenzi,controlPreBersani)
setwd("R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/7categorie")
write.csv(file="controlPre.csv",control)
testoBersaniRenzi=read.csv(".../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/
        +7categorie/controlPre.csv",stringsAsFactors=FALSE)
```

ora bisogna scrivere i singoli file, con all'interno il testo ridotto.
estraiamo i nomi dei file.
nomefileTXT = testoBersaniRenzi$FILENAME[1:1000]
nomefileTXT
nomefileTXTB = testoBersaniRenzi$FILENAME[1001:2000]
nomefileTXTB
Scriviamo i singoli file per Renzi
nomefile = gsub(".txt", "", nomefileTXT, perl=TRUE)
nomefile
numerofileRENZI = as.numeric(nomefile)
numerofileRENZI
textRENZI = dbSoloRenzi$text[numerofileRENZI]
head(textRENZI)
sapply(numerofileRENZI,function(id) cat(dbSoloRenzi$text[id],file=paste(id,sep="",".txt")))
Scriviamo i singoli file per Bersani.
nomefileB = gsub(".txtB", "", nomefileTXTB, perl=TRUE)
nomefileB
numerofileBERSANI = as.numeric(nomefileB)
numerofileBERSANI
textBERSANI = dbSoloBersani$text[numerofileBERSANI]
head(textBERSANI)
sapply(numerofileBERSANI,function(id) cat(dbSoloBersani$text[id],
+ file=paste(id,sep="",".txtB")))
# Leggo il file controlPre.csv
x=read.csv(".../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/
+7categorie/controlPre.csv", stringsAsFactors=FALSE)
str(x)
attach(x)
#creo un data frame con solo le prime tre colonne del file controlPre.txt.
control = data.frame(FILENAME,TRUTH,TRAININGSET)
str(control)
write.table(control, file="control.txt",row.names=FALSE, sep=",", quote=FALSE)
library(ReadMe)
library(quadprog)
oldwd <- getwd()

```
oldwd
# impostare la WD .../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/7categorie.
setwd("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/MONA")
getwd()
undergrad.results <- undergrad(sep = ",")
undergrad.results
undergrad.preprocess <- preprocess(undergrad.results)
undergrad.preprocess
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),
        +length(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))

readme.results <- readme(undergrad.preprocess)
readme.results
readme.results$est.CSMF
readme.results$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)
par(mar=c(1,1,1,1), xpd=TRUE)
plot(albero)
text(albero,all=TRUE,use.n=TRUE)
# Stima dell'errore
plotcp(albero)
printcp(albero)
```

Stima nel TestSet

new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])

head(new.stem)

potatura=prune(albero, cp=0.09)

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 = sum((predetti-reali)^2)

scartiReadMeAlbero=sum((media - reali)^2)

B.4 *Script Bersani Renzi categorie unite, -1,0,1,3.*

Una buona parte del codice è già spiegata in A.4.3,A.4.2 e A.4.1

```
rm(list=ls())
load("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/datiFabio.Rdata")
ls()
idBersani=grep("bersani", db$text)
length(idBersani)
idRenzi= grep("renzi", db$text)
length(idRenzi)
# unione dei due insiemi idBersani e idRenzi.
unioneBersaniRenzi= union(idBersani, idRenzi)
length(unioneBersaniRenzi)
# intersezione tra i due insiemi.
intersezioneBersaniRenzi= intersect(idBersani,idRenzi)
length(intersezioneBersaniRenzi)
# differenza tra unione ed intersezione, dall'unione tolgo gli elementi dell'intersezione.
idBersaniRenzi= setdiff(unioneBersaniRenzi,intersezioneBersaniRenzi)
length(idBersaniRenzi)
dbBersaniRenzi=db[idBersaniRenzi,]
head(dbBersaniRenzi)
idSoloBersani=idBersaniRenzi[1:2353]
dbSoloBersani = db[idSoloBersani, ]
head(dbSoloBersani)
idSoloRenzi = idBersaniRenzi[2354:6405]
dbSoloRenzi = db[idSoloRenzi,]
head(dbSoloRenzi)
controlPreRenzi=read.csv("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/
        +PrimarieDiSinistra/PostRenzi/controlPre.csv")
controlPreBersani=read.csv("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/
        +PrimarieDiSinistra/PostBersani/controlPre.csv")
control = rbind(controlPreRenzi,controlPreBersani)
setwd("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra
        +classificazioneUnica")
write.csv(file="controlPre.csv",control)
# ELIMINARE LA PRIMA COLONNA, A
```

```
testoBersaniRenzi=read.csv("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/
        +PrimarieDiSinistra/classificazioneUnica/controlPre.csv",stringsAsFactors=FALSE)
nomefileTXT  = testoBersaniRenzi$FILENAME[1:1000]
nomefileTXT
nomefileTXTB = testoBersaniRenzi$FILENAME[1001:2000]
nomefileTXTB
# Renzi
nomefile = gsub(".txt", "", nomefileTXT, perl=TRUE)
nomefile
numerofileRENZI = as.numeric(nomefile)
numerofileRENZI
textRENZI = dbSoloRenzi$text[numerofileRENZI]
head(textRENZI)
sapply(numerofileRENZI,function(id) cat(dbSoloRenzi$text[id],file=paste(id,sep="",".txt")))
# Bersani
nomefileB = gsub(".txtB", "", nomefileTXTB, perl=TRUE)
nomefileB
numerofileBERSANI = as.numeric(nomefileB)
numerofileBERSANI
textBERSANI = dbSoloBersani$text[numerofileBERSANI]
head(textBERSANI)
sapply(numerofileBERSANI, function(id) cat(dbSoloBersani$text[id],
        +file=paste(id,sep="",".txtB")))
# Unifichiamo le classificazioni portandole tutte a -1,0,1,3 quindi trasformiamo -2 in -1, 20 in 0
# e 2 in1, la categoria 3 viene unita automaticamente.
testoBersaniRenzi=read.csv("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/
        +PrimarieDiSinistra/classificazioneUnica/controlPre.csv",stringsAsFactors=FALSE)
classificazione = testoBersaniRenzi$TRUTH
classificazione = as.numeric(gsub("20","0",classificazione))
classificazione = as.numeric(gsub("-2","-1",classificazione))
classificazione = as.numeric(gsub("2","1",classificazione))
attach(testoBersaniRenzi)
TRUTH = classificazione
controlPre = cbind(FILENAME,TRUTH,TRAININGSET,message)
write.csv(file="controlPre.csv",controlPre)
```

```
detach(testoBersaniRenzi)
# ELIMINARE LA PRIMA COLONNA, A, SUL FILE CONTROLPRE.CSV.
x=read.csv("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/
PrimarieDiSinistra/classificazioneUnica/controlPre.csv",stringsAsFactors=FALSE)
attach(x)
str(x)
control = data.frame(FILENAME,TRUTH,TRAININGSET)
str(control)
write.table(control, file="control.txt",row.names=FALSE, sep=",", quote=FALSE)
library(ReadMe)
library(quadprog)
oldwd <- getwd()
oldwd
# impostare la WD  .../R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra/
# classificazioneUnica
setwd("/home/fabio/R/x86_64-pc-linux-gnu-library/2.14/ReadMe/PrimarieDiSinistra
        +classificazioneUnica")
undergrad.results <- undergrad(sep = ",")
undergrad.results
undergrad.preprocess <- preprocess(undergrad.results)
undergrad.preprocess
cbind(length(undergrad.results$trainingset),length(undergrad.results$testset),
        +length(undergrad.preprocess$trainingset),length(undergrad.preprocess$testset))
readme.results <- readme(undergrad.preprocess)
readme.results
readme.results$est.CSMF
readme.results$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)

par(mar=c(1,1,1,1), xpd=TRUE)

plot(albero)

text(albero,all=TRUE,use.n=TRUE)

Stima dell'errore

plotcp(albero)

printcp(albero)

Stima nel TestSet

new.stem=(undergrad.preprocess[["testset"]][,-c(1,2,3)])

head(new.stem)

potatura=prune(albero, cp=0.09)

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 = sum((predetti-reali)^2)

scartiReadMeAlbero=sum((media - reali)^2)

BIBLIOGRAFIA

- J. Hopkins, Daniel, King, Gary. 2010. "*A Method of Automated Nonparametric Content Analysis for Social Science.*" *American Journal of Political Science* 54, no. 1: 229–247.
- J. Hopkins Daniel, King Gary, Knowles Matthew e Melendez Steven "*ReadMe: Software for Automated Content Analysis.*"
- Chiogna Monica, Pauli Francesco "*TECNICHE STATISTICHE DI CLASSIFICAZIONE.*"
- http://voicesfromtheblogs.com/.
- R Development Core Team. 2008. "*A Language and Environment for Statistical Computing.*" http://www.R-project.org.
- Masarotto Guido, M. Iacus Stefano. "Laboratorio di statistica con R."
- Hand, David. J. 2006. "*Classifier Technology and the Illusion of Progress.*" *Statistical Science* 21(1):1-14.