Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Laurea Magistrale in Informatica

SENTIMENT ANALYSIS IN TWITTER

Tesi di Laurea Magistrale in Data Mining

Relatore:
Chiar.mo Prof.
Danilo Montesi

Presentata da:
Damiano Melloncelli

Sessione III
Anno Accademico 2012-2013

A Monica, Achille e Silvia

Bologna, 19 Marzo 2014

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

Capitolo 1

Introduzione

Il lavoro svolto durante questa tesi di laurea mira alla progettazione e allo sviluppo di un sistema di analisi testuale in grado di rilevare e classificare opinioni, sentimenti e preferenze espresse dagli utenti sul *social network* Twitter. Più in generale l'idea è quella che analizzando un breve testo, un *tweet* nel nostro caso, il sistema sviluppato sia in grado di determinare il sentimento generale che esso esprime; in altre parole si cerca di determinare l'orientamento (giudizio, pensiero, sentimento o opinione) dell'individuo che ha creato il messaggio.

1.1 Motivazioni

Negli ultimi anni, con la crescita esponenziale nell'uso dei *social media* (recensioni, forum, discussioni, blog e social network), le persone e le aziende utilizzano sempre più le informazioni (opinioni e preferenze) pubblicate in questi mezzi per il loro processo decisionale. Tuttavia, il monitoraggio e la ricerca di opinioni sul Web da parte di un utente o azienda risulta essere un problema molto arduo a causa della proliferazione di migliaia di siti; in più ogni sito contiene un enorme volume di testo non sempre decifrabile in maniera ottimale (pensiamo ai lunghi messaggi di forum e blog). Inoltre, è anche noto che l'analisi soggettiva delle informazioni testuali è passibile di notevoli distorsioni, ad esempio, le persone tendono a prestare maggiore attenzione e interesse alle opinioni che risultano coerenti alle proprie attitudini e preferenze. Risulta quindi

necessario l'utilizzo di sistemi automatizzati di *Opinion Mining*, per superare pregiudizi soggettivi e limitazioni mentali, al fine di giungere ad una metodologia di *Sentiment analysis* il più possibile oggettiva.

1.2 Sentiment Analysis

Con *Sentiment analysis* (anche nota con il nome di *Opinion Mining*) si indica l'insieme delle tecniche e procedure atte allo studio e l'analisi di informazioni testuali, al fine di rilevare valutazioni, opinioni, atteggiamenti ed emozioni relative ad una certa entità (prodotto, persona, argomento, ecc). Questo tipo di analisi ha evidenti ed importanti applicazioni in campo politico, sociale ed economico. Ad esempio, un'azienda potrebbe essere interessata a conoscere le opinioni dei consumatori relative ai propri prodotti. Ma anche i potenziali compratori di un determinato prodotto o servizio saranno interessati a conoscere l'opinione e l'esperienza di qualcuno che ha già acquistato o utilizzato il prodotto. D'altro canto anche un personaggio pubblico (politica, spettacolo, sport) potrebbe essere interessato a sapere cosa la gente pensa di lui. Immaginiamo un personaggio politico, che voglia sapere cosa la gente pensa del suo operato, al fine di monitorare e controllare il consenso per la sua prossima eventuale rielezione. Naturalmente esistono già strumenti per la rilevazione dei consensi e delle opinioni (sondaggi e indagini statistiche); ma tramite tecniche di *Opinion Mining* abbiamo costi di rilevazione nettamente inferiori e in molti casi molta più autenticità informativa: infatti, le persone non sono obbligate ad esprimere opinioni (come in sondaggi e interviste), al contrario, queste fluiscono liberamente senza alcuna costrizione.

1.3 Opinion o Sentiment?

Indispensabile è capire se esiste una differenza tra *opinione* e *sentimento*. Iniziammo dalle basi: che cos'è un opinione? E un sentimento? Il dizionario italiano ci risponde così:

> ***Opinione:*** *concetto che una o più persone si formano riguardo a particolari fatti, fenomeni, manifestazioni, quando, mancando un criterio di certezza assoluta per*

giudicare della loro natura (o delle loro cause, delle loro qualità, ecc.), si propone un'interpretazione personale che si ritiene esatta e a cui si dà perciò il proprio assenso.[1]

Sentimento: *ogni forma di affetto, di impulso dell'animo, di movimento psichico, di emozione, sia che rimangano chiusi entro l'animo della persona stessa, sia che si rivolgano e proiettino verso gli altri, verso il mondo esterno; modo di pensare e di sentire, considerato come parte del carattere di una persona, come complesso delle inclinazioni al bene o al male, come guida del comportamento morale.*[2]

Definizioni di questo genere, sono perfette dal punto di vista linguistico ma troppo astratte e poco precise per una trattazione computazionale del problema. Inoltre le due definizioni sopra differiscono per molti aspetti, ma dal nostro punto di vista (il punto di vista della *sentiment analysis)* il termine opinione ed il termine sentimento indicano la stessa identica cosa. Proviamo a definire il problema in modo più formale [1].

Indichiamo con il termine **entità** un prodotto, servizio, persona, evento o organizzazione. In particolare l'entità e è associata ad una coppia (T, W) dove T rappresenta i componenti o aspetti di e, mentre W è l'insieme degli attributi di e. Ad esempio, un particolare telefono cellulare è un entità; esso è formato da un insieme di componenti (batteria, cover, schermo, ecc.) e possiede un insieme di attributi (risoluzione, qualità della voce, peso, ecc.)

Definiamo un *opinion* (o un *sentiment*) come una quintupla, $(e_i, a_{ij}, oo_{ijkl}, h_k, t_l)$, dove:

- e_i rappresenta l'entità,
- a_{ij} è un aspetto di e_i,
- oo_{ijkl} rappresenta l'orientamento dell'opinione riguardo l'aspetto a_{ij} dell'entità e_i,
- h_k ci dice chi esprime l'opinione (*opinion holder*),
- t_l ci dice quando h_k ha espresso l'opinione (tempo).

Si noti che l'opinione oo_{ijkl} può essere positiva, negativa o neutrale, oppure può essere espressa con differenti livelli di intensità (ad esempio da 1 a 5 stelline).

[1] http://www.treccani.it/vocabolario/opinione/

[2] http://www.treccani.it/vocabolario/sentimento/

1.4 Problematiche sentiment analysis

Rispetto ai problemi di *data mining* classici, come la *topic detection*, troviamo nella *sentiment analysis* difficoltà molteplici in termini di efficacia. Questo è dovuto soprattutto alla sottile distinzione che esiste tra sentimento positivo e negativo (distinzione che risulta spesso difficile anche per un essere umano). Non sempre le opinioni sono espresse tramite l'uso di *opinion words*, in molti casi entrano in gioco altri artefici linguistici come le figure retoriche. Prendiamo come esempio questa frase [1]:

"*Within a month, a valley formed in the middle of the my mattress.*"

L'opinione che l'autore di questa frase ha del proprio materasso è chiaramente negativa, ma essa non è legata a nessuna parola che indica realmente un sentimento. Si tratta di una metafora, il fatto che si sia formata una "valle" significa chiaramente che il materasso è di pessima qualità; questo paragone è banalmente comprensibile per un essere umano ma molto meno chiaro per un classificatore automatico. Proviamo poi a pensare ad una frase contenente ironia o sarcasmo, dove l'interpretazione del significato è strettamente soggettiva a tal punto che, anche due esseri umani, potrebbero trovarsi in disaccordo sul reale sentimento che essa esprime.

Altre difficoltà sono dovute all'uso, sempre più comune in rete, di espressioni non formali e gerghi (*slang*) non appartenenti al vocabolario proprio di una lingua. Questi termini spesso vengono utilizzati in maniera intensiva per esprimere una particolare opinione o un certo stato d'animo; ad esempio il termine "**lol**" indica allegria ed euforia (letteralmente sonora risata). Secondo il dizionario online *Urban Dictionary*[3], il numero di parole, nella sola lingua inglese definite *slang*, supera i sette milioni.

Ulteriori problematiche sono dovute al dominio di definizione del problema; in particolare si nota che i sentimenti espressi da una parola sono spesso dipendenti dal *topic* dell'argomento. Osserviamo questa frase come esempio [1]:

"*It's quite!*"

Essa esprime un opinione positiva se stiamo parlando del motore di un'automobile, ma indica una critica se si parla dell'altoparlante di un telefono. Non è per niente facile

[3] http://www.urbandictionary.com/

generalizzare il problema, anzi è necessario in molti casi avere un'insieme di *features* specifiche per ogni ambito di riferimento.

Concludiamo con il problema *spam*; possiamo classificare al meglio delle nostre possibilità, utilizzare tutte le tecniche possibili per migliorare l'efficenza del nostro sistema di classificazione ma non potremmo mai sapere se il sentimento che rileviamo è realmente vero e sincero oppure è frutto di speculazioni personali o economiche. Esistono comunque tecniche che mirano all'individuazione degli utenti *spammer* tramite metodologie di analisi del testo [2] e studio dello stile comportamentale degli utenti [9].

1.5 Organizzazione del lavoro

Presentiamo di seguito una sintesi del contenuto dei capitoli che seguiranno.

Capitolo 2 Il secondo capitolo comprende una panoramica sullo stato della ricerca sulla *sentiment analysis*; vengono presi in rassegna alcuni lavori valutandone pro e contro.

Capitolo 3 Proseguendo viene introdotto il problema della classificazione testuale con analisi di tecniche e problematiche. Nella seconda parte del capitolo vengono descritti il classificatore Bayesano *Naive* e i metodi di valutazione di un sistema di classificazione.

Capitolo 4 Il quarto capitolo presenta il progetto di tesi. La prima parte introduce il *social network* Twitter descrivendone caratteristiche e punti forti; la seconda parte comprende una trattazione teorica del progetto *Sentiment Analysis in Twitter*, illustrando tecniche utilizzate e problematiche riscontrate.

Capitolo 5 Proseguendo viene descritto l'ambiente di programmazione R, illustrando le caratteristiche principali del linguaggio.

Capitolo 6 Nel sesto capitolo troviamo l'implementazione del progetto di tesi; vengono descritte le scelte implementative e vengono mostrati gli *script* scritti in codice R. Vengono inoltre discussi i risultati sperimentali della fase di *test* del sistema.

Capitolo 7 Si prosegue con un caso studio, cercando di dimostrare caratteristiche e potenzialità del nostro sistema di classificazione.

Capitolo 8 Nell'ottavo e ultimo capitolo si traggono le conclusioni, mettendo in evidenza aspetti positivi e negativi del sistema di classificazione sviluppato e indicando possibili miglioramenti e sviluppi futuri.

Capitolo 2

Related work

La ricerca nel campo della *sentiment analysis* ha subito, in pochi anni una repentina accelerazione; i lavori pubblicati negli ultimi dieci anni sono molteplici. In [1] (*A survey of opinion mining and sentiment analysis*) gli autori propongono una panoramica estesa su tutte le tecniche per la rilevazione e la *detection* di opinioni e sentimenti all'interno di documenti testuali. In particolare il lavoro si concentra su i vari processi di elaborazione del linguaggio naturale che vengono utilizzati nei procedimenti di estrazione e sulle problematiche dovute alla natura intrinsecamente soggettiva del linguaggio naturale, concludendo con un'analisi sulle procedure inerenti alla *spam detection*. Sempre in [1] gli autori cercano di definire in modo formale l'entità "opinione".

In [2] (*Opinion Spam and Analysis*) gli autori ci mettono in guardia sul problema *Spam*. Nelle prime righe dell'articolo leggiamo:

"there is no quality control,
anyone can write anything on the Web."

Sempre nello stesso lavoro viene illustrato un caso studio relativo ad un *set* di *review* tratte da Amazon[1], sul quale gli autori cercano di determinare quali siano veritiere e quali siano fasulle (*spam detection*).

In [3] (*Thumbs up? Sentiment Classification using Machine Learning Techniques*) si prende in esame un *set* di recensioni di films; su di esso, gli autori, eseguono esperimenti di *Opinion Mining* utilizzando tecniche di *Machine Learning*.

[1] http://www.amazon.com/

In [4] (*Sentiment Analysis and Subjectivity*) si danno le basi per la classificazione dei documenti testuali; gli autori dividono i documenti di testo in due grandi gruppi: fatti e opinioni. I fatti sono espressioni indicanti caratteristiche oggettive di entità e rispettive proprietà. Le opinioni sono espressioni soggettive che indicano sentimenti, pensieri e valutazioni personali riguardanti una certa entità e rispettive proprietà. La *sentiment analysis* si riduce, in minimi termini, ad un problema di classificazione. Dato un certo documento, sono necessarie due fasi di classificazione testuale:

- *(soggettività)* distinguiamo i testi che possiedono, oppure no, contenuti soggettivi. Una frase oggettiva presenta informazioni riguardo il mondo (fatti), mentre una frase soggettiva esprime credenze, pensieri o opinioni. Ad esempio la frase: "*dark covers for phones , only 6$. please visit www.myblackcover.com*" è evidentemente oggettiva, infatti non esprime nessuna opinione ma si limita ad esporre un fatto,

- *(polarità)* nel caso in cui il testo fornisca un informazione soggettiva, ne rileviamo il sentimento espresso (positivo, negativo o neutrale). Ad esempio la frase: "*Armageddon is a beautiful and moving film; every time I watch it I get excited*" è soggettiva ed esprime evidentemente un opinione positiva.

Concludendo, non dobbiamo dimenticare quanto i fattori spazio-temporali siano importanti in un processo di *Opinion Mining*. In [5] (*Spatio-Temporal Keyword Queries in Social Networks*), gli autori propongono un modello di *queries* su *social networks* in cui, alla metodologia classica di ricerca, vengono integrati aspetti relativi allo spazio ed al tempo.

2.1 Opinion mining in Twitter

Da qualche anno il campo dell' *opinion mining* all'interno delle reti sociali, Twitter in particolare, ha avuto grande espansione; esistono infatti servizi in rete che ci permettono di recuperare tweets inerenti ad un certo argomento (ricerca con *key-word*) determinandone la relativa opinione espressa (sentimento positivo, negativo o neutro). Uno di questi è *Sentiment140*[2] sviluppato da un team di ricerca della Stanford University. Inoltre molti

[2]http://www.sentiment140.com/

lavori di ricerca sono stati pubblicati: in [10] (*Automatic detection of political opinions in Tweets*) gli autori cercano di determinare l'orientamento politico degli utenti di Twitter nel periodo precedente alle elezioni inglesi del 2010; mentre in [11] (*Twitter Sentiment Classification using Distant Supervision*) si cerca di determinare l'opinione espressa dai *tweets* attraverso l'uso di algoritmi di apprendimento supervisionato associati alla rilevazione delle *emoticon*, ottenendo risultati incoraggianti con un'*accuracy*[3] pari al 81.3% (algoritmo Naive Bayes[4]), 80.5% (algoritmo Maximum Entropy[5]), 82% (algoritmo Support Vector Machine[6]). Risultati molto simili sono riscontrabili anche in [7] (*Twitter as a Corpus for Sentiment Analysis and Opinion Mining*). Si nota che i risultati migliori si hanno con una metodologia di classificazione di tipo "*unigrams*"[7] (vedi sezione 3.1.1 per approfondimento), considerando cioè un solo termine per volta durante la classificazione (in [11] e in [7]).

[3] generalmente si intende il rapporto tra il numero di documenti classificati correttamente e il numero totale di documenti da classificare

[4] vedi sezione 3.2 per approfondimenti

[5] http://en.wikipedia.org/wiki/Maximum_entropy_classifier

[6] http://en.wikipedia.org/wiki/Support_vector_machines

[7] http://en.wikipedia.org/wiki/N-gram

Capitolo 3

Classificazione testuale

Il capitolo introduce nella prima parte il problema della classificazione testuale descrivendone tecniche, problematiche e soluzioni adottate in letteratura, mentre nella seconda parte descrive un particolare tipo di classificatore (*Naive Bayes*) e spiega come avviene la valutazione delle prestazione in un sistema di classificazione.

3.1 La classificazione

La classificazione testuale o *text categorization* è un problema di *data mining* che consiste nell'assegnare (classificare) etichette a testi scritti in linguaggio naturale. Gli approcci possibili al problema sono rappresentati da due tecniche di intelligenza artificiale: *rule based* e *machine learning* (o apprendimento automatico).

Il primo consiste nella formazione di un gruppo di regole per la creazione di un sistema esperto [6] (*INTELLIGENZA ARTIFICIALE Un approccio moderno*) adatto alla risoluzione di un determinato problema. Nell'ambito della classificazione testuale le *rules*, che sono parole chiave ed espressioni regolari, combinate tramite opportuni operatori logici, vengono utilizzate nel processo di inferenza per l'etichettatura dei testi.

3.1.1 Machine learning

Con lo sviluppo delle tecniche statistiche applicate all'intelligenza artificiale (inizio anni 90), il secondo approccio ha riscosso molto successo, sostituendosi nella maggio-

ranza delle applicazione al paradigma *rule based*. Parliamo quindi di sistemi che sono in grado di acquisire e migliorare le proprie competenze valutando l'esperienza passata. In pratica gli algoritmi di apprendimento automatico prendono in *input* un *set* di testi e restituiscono in *output* un modello generale per la classificazione di nuovi testi. Nel panorama dell'apprendimento automatico distinguiamo due categorie [8] (*Foundation of statistical natural language processing*):

- *supervised learning* dove il sistema riceve, in una fase iniziale di *training*, un insieme di dati gia classificati ed etichettati. Da questo acquisisce conoscenza ed esperienza per classificare i dati successivi;

- *unsupervised learning* dove il sistema riceve, nella fase di *training*, una serie di dati non etichettati che verranno classificati sulla base di caratteristiche comuni. Al contrario dell'apprendimento super-visionato, le classi non sono note a priori ma devono essere apprese automaticamente.

Portiamo la nostra attenzione sulle tecniche di apprendimento super-visionato e definiamo formalmente il problema. Il nostro testo da classificare è definito da un insieme di m valori reali (anche detti *features*); ad ogni istanza del problema bisogna associare un etichetta scelta in un insieme di possibili etichette E; l'algoritmo richiede in input il *training set* $S = \{\langle x_i, y_i \rangle \, | i \in (0 \dots n)\}$ dove $x_i \in R^m$ (esempio di oggetto da classificare) e $y_i \in E$ (etichetta ad esso associata); alla fine viene prodotta una funzione di modello $\hat{y}_i = f(x_i)$ che deve essere in grado di massimizzare il numero di oggetti classificati correttamente, ovvero per cui $\hat{y}_i = y_i$; l'accuratezza A dell'algoritmo si misura proprio tramite il rapporto tra il numero di oggetti correttamente classificati e il numero totale di quelli che sono stati dati in pasto all'algoritmo.

Nei problemi di *sentiment analysis* l'insieme E è solitamente discreto, in particolare si parla spesso di classificazione binaria (opinione positiva o negativa). Mentre lo spazio necessario alle istanze del problema è R^m. Ma come si trasforma un documento testuale in un *array* di valori numerici (insieme delle *features*)?

3.1.2 Estrazione delle features

Il procedimento che trasforma un testo in un vettore di numeri è detto *feature extraction*; si noti che la similarità tra due *array* determina la similarità tra i due testi da qui questi vengono derivati. Definiamo T $(t_0, \ldots, t_m)$ come un insieme di termini, ogni documento d è rappresentato da un *array* $\langle w_0, w_1, \ldots, w_m \rangle$ in cui l'elemento w_i per $i \in (0 \ldots m)$, ci dice quanto il termine t_i contribuisce al significato del testo o, in altre parole, quanto il termine t_i definisce l'opinione espressa dal testo.

Termini : la scelta più facile e immediata è quella di associare i termini con le parole (*bag of words*) [8]. Le parole possono essere ad esempio prese da un dizionario di *opinion words*[1]. In [1] si mostra come da un piccolo gruppo di *opinion words*, tramite l'utilizzo iterativo di un dizionario dei sinonimi e contrari, si può giungere ad un vero e proprio vocabolario di espressioni che indicano opinioni e stati d'animo. Oppure i termini possono anche essere identificati con le parole dell'insieme dei documenti presi in riferimento, ad esempio tutte le parole che compaiono almeno un certo numero di volte nei documenti. Scelte più raffinate includono l'associazione dei termini a gruppi di parole o n-grammi [8]. Un n-gramma è una sotto-sequenza contigua di n caratteri presente nella sequenza che stiamo analizzando; ad esempio i 4-grammi della parola **eurozona** sono: **euro**, **uroz**, **rozo**, **ozon** e **zona**.

Pesi (w) : una volta definito un termine è necessario calcolarne il peso w all'interno di un documento. L'approccio più banale è porre $w = 0$ se il termine non è presente nel documento e $w = 1$ altrimenti. Un altro metodo è l'utilizzo della *term frequency* o `tf` (quante volte il termine compare nei documenti) o dell'*inverse document frequency* o `idf` (reciproco del numero dei documenti in cui compare il termine). Nel caso della *sentiment analysis* molto spesso è proprio il termine che compare con maggiore frequenza ad avere maggiore importanza, quindi una pesatura che utilizza il metodo `idf` non avrebbe grossi risultati.

[1] Parole, termini e modi di dire appartenenti al linguaggio naturale che solitamente indicano sentimenti, opinioni e stati d'animo

3.1.3 Riduzione delle features

Molto spesso accade che, alla fine del processo di estrazione delle *features* e di creazione del dizionario, l'insieme dei termini T sia troppo esteso; questo fatto può causare sprechi computazionali dell'algoritmo e problemi di *overfitting* con una conseguente cattiva classificazione. Si parla di *overfitting* quando un modello statistico si adatta ai dati osservati (il campione) usando un numero eccessivo di parametri. Da notare che un modello assurdo e sbagliato può adattarsi perfettamente se è abbastanza complesso rispetto alla quantità di dati disponibili[2]. Esistono quindi tecniche per la riduzione delle *features* che vanno ad eliminare i termini meno significativi dall'insieme T. La prima idea è valutare il fattore di *sparsity* dei termini, cioè con che frequenza documentale compaiono; si può ad esempio scartare tutti i termini che compaiono un numero di volte inferiore ad una certa soglia (un termine che compare in pochi documenti non sarà significativo ai fini della classificazione). Oppure, al contrario scartare quei termini che hanno frequenza di apparizione troppo alta (come le cosiddette *stopwords*[3]) che non danno nessun contributo significativo ai fini della classificazione.

Altre tecniche per la selezione dei termini in T usano metodologie legate all'elaborazione del linguaggio naturale. Ricordiamo lo *stemming*, in cui andiamo a sostituire ad ogni parola il proprio tema (*steam*). Ad esempio le parole: "andai", "andare", "andiamo", "andò" vengono tutte sostituite con "and". Lo *stemming* viene eseguito solitamente tramite programmi basati su dizionario: essi infatti conoscono le *words* di una lingua e sono in grado di ricavarne il tema. Altre metodologie, sempre basate su NLP, utilizzano dizionari dei sinonimi e contrari per cercare di non mantenere più *words* con lo stesso significato.

3.1.4 Lessico e semantica

L'analisi semantica e lessicale del linguaggio può venirci in aiuto per migliorare le prestazioni dei sistemi di classificazione. Oltre alle già citate *opinion words*, esistono altre caratteristiche del linguaggio che possono essere tenute in considerazione in un sistema di *opinion mining*.

[2] http://it.wikipedia.org/wiki/Overfitting

[3] Vocaboli largamente usati in una lingua come articoli, pronomi, congiunzioni, ecc.

Part of speech : alcune parti del discorso possono essere ottimi indicatori di *opinion*. Ad esempio gli aggettivi, che spesso sono un importante indicatore di opinione, possono essere trattati come *features* particolari. Anche i modi di dire (*opinion phrases*), di una particolare lingua, sono evidentemente utili per la rilevazione del *sentiment*.

Negazioni e congiunzioni : le clausole di negazione sono molto importanti perché solitamente, all'interno di un periodo, esprimono un cambiamento dell'orientamento espresso dalla *opinion word*. Nel periodo "*It's not nice!*" l'*opinion word* rilevata (*nice*) esprime, senza ombra di dubbio, un sentimento positivo, ma l'applicazione della negazione muta e inverte il sentimento espresso dalla *word* e, conseguentemente, della frase.

Sentiment consistency : in un periodo complesso spesso è possibile che vengano manifestate più opinioni in accordo o non. La teoria della *sentiment consistency* [1] afferma che in un periodo in cui sono espresse due opinioni, separate da una clausola di congiunzione o contrapposizione, è sufficiente rilevare una sola opinione per intuire il significato anche della seconda. La clausola di congiunzione esprime mantenimento di opinione, ad esempio nella frase "*my car is cheap* ***and*** *comfortable*" l'opinione positiva perdura prima e dopo la clausola. Invece, nella frase "*my car is cheap* ***but*** *ugly*" l'opinione positiva si trasforma in negativa dopo la clausola.

3.1.5 Modelli di classificatori

Esistono diversi tipi di classificatori che differiscono prevalentemente per come viene costruita la funzione di classificazione. Di seguito vengono descritti in maniera sintetica alcuni modelli di classificatori:

- **logici:** dove la funzione di classificazione è espressa mediante condizioni logiche sui valori degli attributi (Alberi e Regole di Decisione);
- **matematici:** dove la classificazione è affidata ad una funzione matematica che mappa i parametri in entrata in un elemento del dominio delle classi in uscita (Reti Neurali, *Support Vector Machines*);

- **basati sugli esempi:** in cui vengono memorizzati tutti gli esempi (con classe nota) del *training set*; il sistema classifica poi ogni singolo elemento valutandone la "somiglianza" con gli esempi memorizzati (*Nearest Neighbor*);

- **statistici:** dove, nella fase di *training*, vengono memorizzati i parametri delle varie distribuzioni di probabilità relative alle classi ed agli attributi; la classificazione avviene tramite la valutazione delle probabilità degli attributi (dell'elemento che si vuole classificare) nelle varie classi (*Naive Bayes*).

3.2 Approccio Bayesiano alla classificazione

Il classificatore Bayesiano esegue una classificazione di tipo statistica improntata al calcolo della probabilità delle cause (o probabilità a posteriori), cioè, avvenuto un certo evento si determina la probabilità di quale causa lo abbia scatenato. Il classificatore deriva direttamente dal celeberrimo teorema introdotto dal matematico Thomas Bayes.

3.2.1 Teorema di Bayes

Il teorema deriva da tre leggi fondamentali della probabilità: teorema della probabilità condizionata, teorema della probabilità composta, teorema della probabilità assoluta.

Probabilità condizionata. La probabilità di A condizionata da B è definita come la probabilità che si verifichi l'evento A a condizione che si verifichi anche l'evento B (naturalmente con $P(B) \neq 0$). Enunciamo il teorema in formula (3.1).

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{3.1}$$

Probabilità composta. La probabilità che due eventi A e B si verifichino contemporaneamente è uguale alla probabilità di uno dei due eventi ($P(A)$ o $P(B)$) moltiplicato con la probabilità dell'altro evento condizionato dal verificarsi del primo ($P(A|B)$ o $P(B|A)$). Enunciamo il teorema in formula (3.2).

$$P(A \cap B) = P(B)P(A|B) = P(A)P(B|A) \tag{3.2}$$

Probabilità assoluta. Se $A_1, \ldots, A_n$ formano una partizione dello spazio di tutti gli eventi possibili Ω, (ossia $A_i \cap A_j = \oslash$ con $1 \leq i, j \leq n$ e $\bigcup_{i=1}^{N} A_i = \Omega$) e B un qualsiasi evento dipendente dagli eventi A, allora:

$$P(B) = \sum_{i=1}^{N} (A_i) P(B|A_i) \tag{3.3}$$

Enunciamo infine il teorema di Bayes nella sua forma più semplice:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.4}$$

Dati due eventi legati da una relazione di tipo causa - effetto il teorema cerca di rispondere a questa domanda: dato un certo effetto, qual'è la probabilità che sia stato prodotto da una certa causa?

3.2.2 Classificatore Bayesano Naive

Data una classe di interesse C e un insieme di caratteristiche (o *features*) $F_1, ..., F_n$ di un individuo si vuole conoscere con quale probabilità questo appartiene a C, ossia si vuole conoscere $P(C|F_1, ..., F_n)$. Applicando la regola della probabilità composta (3.2) torniamo alla forma base del teorema di Bayes:

$$P(C|F_1, \ldots, F_n) = \frac{P(C)P(F_1, \ldots, F_n|C)}{P(F_1, \ldots, F_n)} \tag{3.5}$$

Applicando n volte la definizione di probabilità condizionata (3.1) e trascurando il denominatore (definito costante di normalizzazione) abbiamo:

$$P(C|F_1, \ldots, F_n) = P(C)P(F_1|C)P(F_2|C, F_1) \ldots P(F_n|C, F_1, \ldots, F_{n-1}) \tag{3.6}$$

Il generico termine $P(F_i|C, F_1, \ldots, F_{i-1})$ diventa $P(F_i|C)$ per assunzione *Naive* di indipendenza tra i termini. Per teorema della probabilità composta (3.2) abbiamo:

$$P(C|F_1, \ldots, F_n) = P(C)P(F_1|C)P \ldots (F_n|C, F_1) = \prod_{i=1}^{n} P(F_i|C) \tag{3.7}$$

Questo tipo di classificatore si indica come *naive* perché è basato sull'assunzione esemplificativa che tutti gli attributi (*features*) che descrivono una certa istanza sono tra

loro indipendenti data la categoria a cui appartiene l'istanza. Questa affermazione viene detta assunzione del Naive Bayes. Nonostante questa assunzione sia violata nella maggior parte dei problemi reali come, ad esempio, nella categorizzazione del testo, il Naive Bayes si comporta molto bene e risulta essere molto efficace. L'assunzione di indipendenza permette di apprendere separatamente i parametri di ogni attributo, semplificando e snellendo molto l'apprendimento. Il dominio di applicazione del classificatore Naive Bayes riguarda la classificazione di istanze che possono essere descritte mediante un insieme di *features* di cardinalità (n in formula (3.7)) anche molto elevata. Il nostro caso (classificazione di documenti) è uno di quei domini con un grande numero di attributi. Una scelta comune associa gli attributi con le parole del documento e si capisce quindi che il loro numero può diventare molto elevato. Esistono tecniche di apprendimento che riducono la taglia del vocabolario, in modo da avere pochi attributi da gestire. Purtroppo nel campo della categorizzazione pochi sono i termini che non sono rilevanti ai fini della classificazione, quindi un'eccessiva diminuzione porta ad un peggioramento delle prestazioni del classificatore. La soluzione consiste nell'impiegare delle tecniche in grado di elaborare un numero elevato di attributi. I classificatori Bayesiani non sono comunque l'unico approccio alla categorizzazione del testo: ad esempio, esistono altri orientamenti come quello delle Support Vector Machines[4].

3.3 Come valutare un classificatore?

Come riuscire a capire se il nostro classificatore produce dei buoni risultati oppure no? La valutazione prestazionale dei classificatori di testi è effettuata sperimentalmente piuttosto che analiticamente, in quanto una valutazione di tipo analitico non può essere formalizzata (a causa della natura soggettiva del problema della classificazione). La valutazione sperimentale di un classificatore solitamente misura la sua efficacia, ovvero l'abilità di prendere la giusta decisione durante il processo di classificazione. Questa viene solitamente misurata in termini di precisione (π) e *recall* (ρ).

La precisione π_i rispetto alla classe c_i è definita come la probabilità condizionata $P(\Phi^*(d_x, c_i) = T|\Phi(d_x, c_i) = T)$ che un documento d_x (scelto casualmente) sia

[4]http://en.wikipedia.org/wiki/Support_vector_machine

classificato sotto la classe c_i e che questa decisione sia corretta.

Il *recall* ρ_i rispetto alla classe c_i è definito come la probabilità condizionata $P(\Phi(d_x, c_i) = T|\Phi^*(d_x, c_i) = T)$ che un documento d_x (scelto casualmente) necessiti di essere classificato in classe c_i e che questo avvenga durante la classificazione.

Per ogni classe definiamo quattro tipi di documenti:

- $\mathbf{FP}_i$: (falsi positivi) numero di documenti classificati incorrettamente sotto la classe c_i;
- $\mathbf{TP}_i$: (veri positivi) numero di documenti classificati correttamente sotto la classe c_i;
- $\mathbf{FN}_i$: (falsi negativi) numero di documenti non classificati erroneamente sotto la classe c_i;
- $\mathbf{TN}_i$: (veri negativi) numero di documenti non classificati correttamente sotto la classe c_i.

Le stime dei parametri di precisione e *recall* per la classe c_i possono essere ottenute come:

$$\pi_i = \frac{TP_i}{TP_i + FP_i} \tag{3.8}$$

$$\rho_i = \frac{TP_i}{TP_i + FN_i} \tag{3.9}$$

Per ottenere precisione e *recall* del classificatore possiamo procedere in due modi: tramite macro-media o micro-media.

Micro-media: π e ρ sono ottenute sommando tutte le decisioni individuali (μ indica la micro-media)

$$\widehat{\pi}^{\mu} = \frac{TP_i}{TP_i + FP_i} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FP_i)} \tag{3.10}$$

$$\widehat{\rho}^{\mu} = \frac{TP_i}{TP_i + FN_i} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|}(TP_i + FN_i)} \tag{3.11}$$

Macro-media: π e ρ sono prima valutate localmente per ogni categoria e poi globalmente tramite la media dei risultati (M indica la macro-media).

$$\widehat{\pi}^M = \frac{\sum_{i=1}^{|C|} \pi_i}{|C|} \tag{3.12}$$

$$\widehat{\rho}^M = \frac{\sum_{i=1}^{|C|} \rho_i}{|C|} \tag{3.13}$$

Esistono altri tipi di misure utilizzate per la valutazione di un classificatore; di seguito definiamo **accuratezza** (formula (3.14)) ed **errore** (formula (3.15)).

$$\widehat{A} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.14}$$

$$\widehat{E} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \widehat{A} \tag{3.15}$$

Capitolo 4

Progetto Sentiment Analysis in Twitter

Il progetto *Sentiment Analysis in Twitter* ha come obbiettivo la costruzione di un sistema automatico per la rilevazione e valutazione delle opinioni espresse sul *social network* Twitter. Formalmente il sistema dovrà essere in grado di:

- acquisire informazione dalla rete (*tweets*);
- classificare le informazioni acquisite tramite la valutazione e l'estrazione del sentimento espresso.

Il capitolo viene suddiviso in due parti; nella prima viene presentata una panoramica sul *social network* Twitter, dando risalto al perché risulti essere un ottimo ambiente per l'analisi sociale, mentre nella seconda parte viene presentato lo schema di studio e realizzazione del progetto di tesi descrivendo tecniche implementative, problematiche e soluzioni adottate.

4.1 Social network: Twitter

Twitter é una piattaforma gratuita di social network e microblogging, si "auto-definisce" così:

Twitter is the best way to connect with people, express yourself and discover what's happening[1].

Ad oggi è una delle reti sociali più utilizzate al mondo. Milioni di utenti ogni giorno lo usano per condividere e visualizzare le più disparate informazioni: opinioni, commenti, news e molto altro.
Twitter nasce nel marzo 2006 dalla Obvious Corporation di San Francisco. Ecco alcuni 'numeri' del social network secondo *S-1*[2]: documento che ufficializza la procedura di *Initial Public Offering* depositato da Twitter nel secondo trimestre del 2013.

- Circa 220 milioni gli utenti attivi;
- Oltre 500 milioni i tweets scambiati ogni giorno;
- Circa 100 milioni gli utenti che accedono ai servizi di Twitter ogni giorno;
- Oltre 250 milioni di dollari di ricavi nei primi 6 mesi del 2013.

4.1.1 Struttura di Twitter

Il servizio è caratterizzato dalla pubblicazione di brevi messaggi. I messaggi, denominati *tweets* sono di soli 140 caratteri; essi possono contenere opinioni, pensieri immagini, video e riferimenti a contenuti multimediali in genere reperibili in Rete.

In Twitter è possibile "seguire" un gruppo di utenti ed essere "seguiti". Seguire un utente significa ricevere e visualizzare tutti i messaggi che esso pubblica. Per ogni profilo utente distinguiamo quindi due gruppi sociali:

- *followers*: insieme degli utenti da cui si è seguiti;
- *following*: insieme degli utenti che si seguono.

Ogni pagina profilo ha una propria *timeline*(figura 4.1) dove compaiono tutti i *tweets* pubblicati dai propri *following*.

[1] https://about.twitter.com
[2] http://www.sec.gov/Archives/edgar/data/1418091/000119312513390321/d564001ds1.htmtoc5640011

A differenza di altri sistemi, Twitter permette la relazione tra due utenti anche in un solo verso, ovvero un utente può seguirne un altro anche se quest'ultimo non lo segue a sua volta.
Twitter nasce come strumento di comunicazione unilaterale in cui qualcuno scrive e qualcun'altro legge, ma nel tempo cambia integrando il concetto di conversazione. Si può rispondere o menzionare un utente attraverso il simbolo @ anteposto al nickname utente. Si può creare un dibattito o una conversazione su un particolare tema aggiungendo il simbolo # anteposto al *topic* del dibattito. Twitter viene sempre più utilizzato da aziende (figura 4.2), personaggi famosi e politici (figura 4.3) come canale di comunicazione e interazione con i propri clienti, i propri fans o i propri elettori.

4.1.2 Perchè Sentiment analysis on Twitter?

Considerando i dati presentati da Twitter in *S-1*[3] ogni secondo poco meno di 6000 tweets vengono pubblicati nella "twittosfera". Risulta facile immaginare l'enorme potenziale informativo di questa mole di dati. Sicuramente più difficile l'estrapolazione, l'interpretazione e la comprensione di questa miriade di informazioni. La difficoltà maggiore è proprio dovuta alla quantità di dati. Per capire meglio il problema si può pensare a questo piccolo esempio. Immaginiamo un enorme piazza affollata di persone; ci piacerebbe sapere cosa pensa la gente relativamente ad un certo argomento; ora immaginiamo di avere un microfono che registra ogni singola frase pronunciata in questa piazza. Alla fine del nostro esperimento avremmo a disposizione milioni di frasi ma solo una piccola percentuale saranno di nostro interesse. Tuttavia Twitter e le reti sociali in genere sono un ottimo ambiente per condurre analisi e indagini di vario tipo e ciò è dovuto soprattutto alla semplicità di utilizzo dello strumento; infatti dopo aver creato il proprio account, l'utente può immediatamente scrivere tutto ciò che gli passa per la mente. Inoltre Twitter è anche condivisione tematica: l'utente interagisce con amici e colleghi, oppure segue l'uscita di un nuovo prodotto *hi-tech* e riporta le proprie opinioni. Altri lo seguono e sullo stesso argomento dicono la loro. Così da questa modalità di libera conversazione possono uscire opinioni ed esperienze personali utili ad esempio per determinare il grado di soddisfazione degli utenti riguardo un determinato prodotto.

[3]Documento di *Initial Public Offering* pubblicato da Twitter (secondo trimestre 2013)

Figura 4.1: Timeline del profilo di Enrico Letta

Figura 4.2: Profilo di Volkswagen Italia"@Volkswagen_IT"

Figura 4.3: Profilo ufficiale di Enrico Letta "@EnricoLetta"

4.1.3 Mining tweets

Twitter è di base un ambiente "pubblico" senza particolari limitazioni, infatti come gia detto permette la lettura e la condivisione dei messaggi anche di utenti non connessi alla propria rete di amici. Ad avvertirci di questo fatto è una delle prime righe dell'Informativa sulla privacy di Twitter:

Quello che dici su Twitter può essere visto istantaneamente in tutto il mondo[4]

Anche per questo motivo è un ottimo ambiente per la ricerca e l'estrazione dei dati. Il recupero dei *tweets* viene effettuato tramite una serie di *API*[5] con le quali è possibile collegarsi ai server di Twitter per scaricare dati. Le *API* nascono per la realizzazione di applicazioni in grado di replicare le funzionalità presenti sulla propria pagina web di Twitter, come, ad esempio, visualizzare i *tweets* degli utenti che si seguono, visualizzare l'insieme dei Followers o Following, ricercare tweet per contenuti, ecc. Esse hanno invece limitazioni per scaricare grandi quantità di dati e, in generale, per ottenere tutti i dati disponibili. Utilizzare le *API* di Twitter si riduce in realtà all'esecuzione di richieste HTTP che restituiscono i risultati in un formato strutturato (JSON nel nostro caso). Le *application program interface* di Twitter sono integrate nella maggior parte dei linguaggi di programmazione e per il nostro progetto utilizziamo un package di *R* (vedi capitolo 5) chiamato *streamR* (vedi sezione 5.2).

Streaming API

Essenzialmente il nostro problema si riduce al recupero dei *tweets* contenenti una o più *keywords* (ricerca per contenuti). Le *streaming API di Twitter* ci mettono a disposizione il metodo *POST statuses/filter*[6]. Esso accetta come parametri una o più *keywords* e fornisce come risultato una lista di *tweets* che fanno *match* con le *words* passate come parametro. Inoltre è possibile specificare come parametro una o più box[7] geografiche per recuperare ad esempio i messaggi provenienti solo da una certa zona (ri-

[4] https://twitter.com/privacy
[5] https://dev.twitter.com/docs/api
[6] https://dev.twitter.com/docs/api/1.1/post/statuses/filter
[7] https://dev.twitter.com/docs/streaming-apis/parameters#locations

cerca geo-localizzata). Il metodo richiede autenticazione tramite il protocollo OAuth[8]. In particolare una volta autenticati si ottiene un token che non ha scadenza e che deve essere usato per tutte le richieste di connessione future.

Tweet

I dati recuperati utilizzando le *streaming API di Twitter* comprendono, oltre al campo testuale del *tweet*, molte altre informazioni aggiuntive. I campi più significativi del dato **tweet**[9] vengono illustrati di seguito:

- *text*: testo del tweet, di massimo 140 caratteri;
- *user*: intero che identifica l'utente che ha postato il messaggio;
- *retweet_count*: numero di volte che il messaggio è stato "*retweettato*";
- *lang*: indica la lingua di scrittura del messaggio;
- *id*: intero che identifica il *tweet*;
- *favorite_count*: numero di volte che il messaggio è stato scelto come *preferito*;
- *created_at*: data di creazione del messaggio (espresso UTC time[10]);
- *coordinates*: cordinate geografiche del lugo da cui il messaggio è stato posta-to(longitudine, latitudine);
- *in_reply_to_user_id*: se il *tweet* corrente è una *reply*, questo campo contiene l'*id* dell'utente che ha postato il *tweet* a cui esso è in risposta;
- *in_reply_to_status_id*: se il *tweet* corrente è una *reply*, questo campo contiene l'*id* del *tweet* a cui esso è in risposta.

[8] https://dev.twitter.com/docs/auth/obtaining-access-tokens
[9] https://dev.twitter.com/docs/platform-objects/tweets
[10] http://en.wikipedia.org/wiki/Unix_time

4.2 Sentiment Anlysis in Twitter: linee guida

Il progetto *Sentiment Anlysis in Twitter* ha come fine lo sviluppo di un sistema *software* in grado di rilevare e classificare il sentimento espresso in brevi testi raccolti da un servizio di *micro blogging*, Twitter nel caso specifico. Per lo sviluppo del progetto si è utilizzato il linguaggio **R** (vedi capitolo 5).

Dominio applicativo

1. Lunghezza testi: la lunghezza massima di un messaggio di Twitter è di 140 caratteri; per cui un *tweet* è formato da una o massimo due frasi.

2. Disponibilità dati: le Twitter *API* consentono la ricerca e il download dei *tweet* tramite *query* con vari tipi di filtri (*keywords*, geografico, linguistico).

3. Linguaggio: il nostro sistema lavora su i soli messaggi scritti in lingua inglese.

4. Emoticons: per una corretta classificazione è necessario considerare l'uso di questi espedienti linguistici; data la poca lunghezza disponibile, l'uso delle emoticons per esprimere sentimenti e stati d'animo è molto diffuso nei *tweet*.

5. Gergo: il cosiddetto *casual language* è molto utilizzato in ambito Twitter; ad esempio le parole *happpyyy, happiie, hap-e, happy,* hanno tutte le stesso significato e vengono usate per denotare felicità.

6. Simboli speciali e URL: molti *tweets* contengono collegamenti WEB e simboli non testuali quali RT, #, @.

4.2.1 Caratteristiche del sistema

Classificatore: la classificazione testuale viene eseguita tramite l'implementazione di un classificatore Naive Bayes. L'estrazione delle *features* è orientata all'approccio *bag of words*; in particolare viene eseguita un associazione di tipo 1:1 tra le *features* e le parole del testo (*tweet*) preso in esame. Per la pesatura del termine si utilizza

il metodo **tf** (*term frequency*), ovvero la frequenza di apparizione del termine all'interno dei documenti. Il dominio delle classi comprende tre possibili categorie: sentimento positivo, sentimento negativo, sentimento neutro (non classificabile).

Riduzione delle features: per ridurre il numero di termini campionati vengono utilizzate le seguenti tecniche: *stemming*, eliminazione *stopwords*, rimozione delle lettere ripetute all'interno delle parole (esempio *looooove* diventerà *loove*, *haaaaaappyyy* diventerà *haappyy*), eliminazione URL, eliminazione di *@username* e *#tag*, trasformazione di tutte le lettere maiuscole in minuscole.

Emoticon: molte volte una *smile* può comunicare sentimenti più di mille parole. Viene effettuata la rilevazione delle emoticons, sostituendo a quelle positive il termine "*posmaremo*" (*positive marker emoticon*) e quelle negative il termine "*negmaremo*" (*negative marker emoticon*).

Filtraggio testuale: l'estrazione delle *features* viene preceduta da una fase di "pulizia" testuale in cui vanno eliminati spazzi ripetuti, punteggiatura ed *RT*.

Tweets: per le fasi di addestramento e verifica del sistema viene utilizzato un *dataset* contenete *tweets* già classificati, in quanto una pre-classificazione "umana" dei dati sarebbe stata molto onerosa. Per il caso studio vengono utilizzati *tweets* recuperati tramite API Twitter. In particolare sono state utilizzate le *streaming* API che ci hanno consentito il filtraggio testuale dei messaggi per *keyWord*; talvolta può accadere che queste funzioni messe a disposizione da Twitter restituiscano più volte lo stesso *tweet*, in quel caso i "doppioni" vengono eliminati, prima della classificazione, tramite il controllo del *tweet-id*.

Geo-localizzazione: una piccola percentuale dei *tweets* recuperati contiene informazione (latitudine e longitudine) per risalire al luogo da cui il messaggio è stato *twettato*. In quei casi le coordinate vengono estratte per eseguire una localizzazione spaziale del sentimento.

Le fasi temporali di progettazione ed esecuzione del progetto di tesi Sentiment Analysis in Twitter sono fondamentalmente due: la prima comprende *training* e *testing* del sistema,

la seconda invece propone un caso studio reale in cui vengono recuperate le informazioni (tramite Twitter API) e su di esse viene fatta classificazione. In figura 4.4 viene mostrato lo schema esemplificativo del progetto Sentiment Analysis in Twitter.

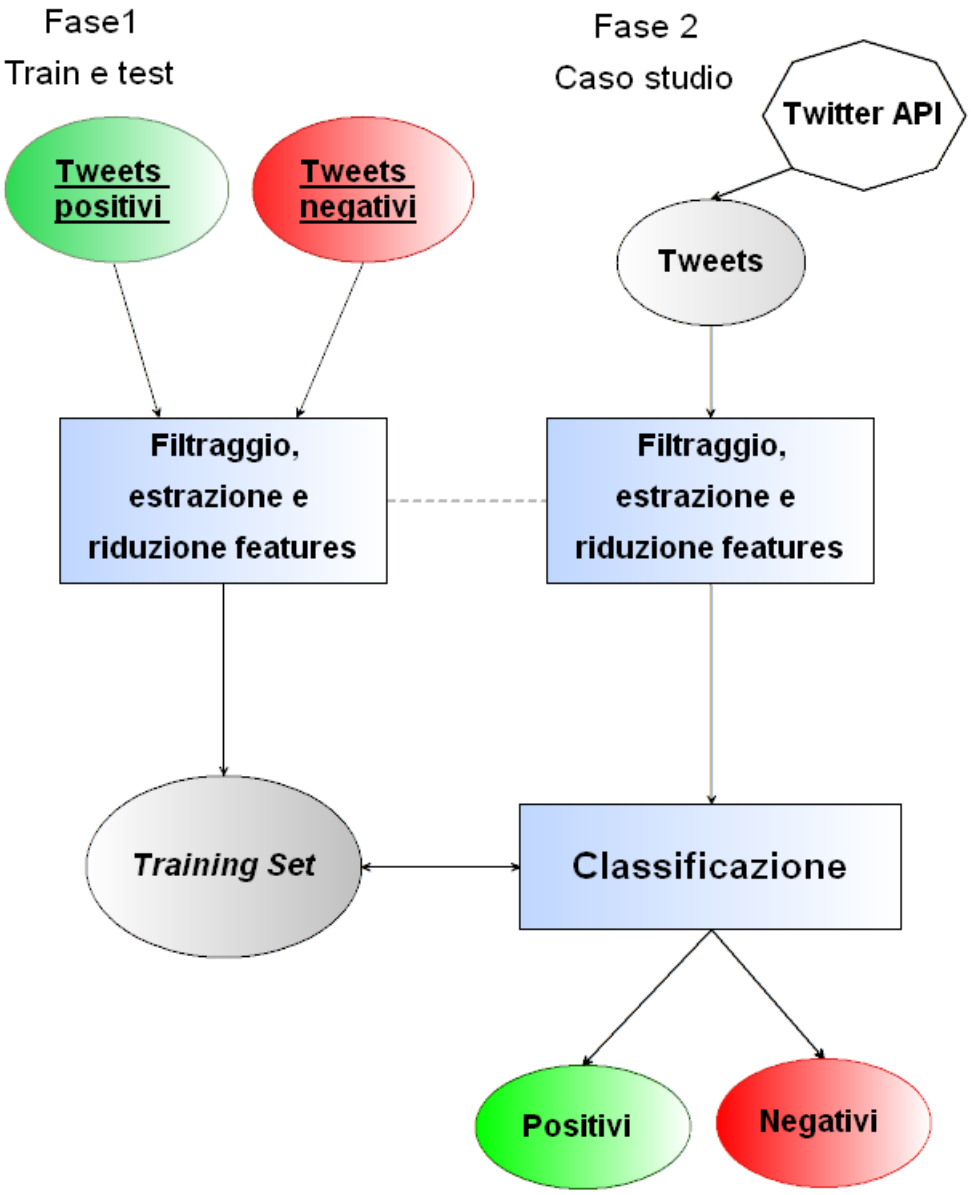

Figura 4.4: Diagramma a blocchi progetto Sentiment Analysis in Twitter

Capitolo 5

R System

Il progetto di tesi è stato sviluppato in linguaggio R[1]; la denominazione di "linguaggio" non è del tutto corretta, esso infatti è un ambiente di sviluppo software specifico per il calcolo statistico e matematico. I creatori di R sono Ross Ihaka e Robert Gentelman, del dipartimento di Statistica dell'università di Auckland, Nuova Zelanda. Nel 1997 si costituisce l'R Core Team, che ha come scopo lo sviluppo e il controllo dei codici sorgenti. R deriva ed è molto simile al linguaggio S[2] (ambiente che è stato sviluppato presso i Bell Laboratories da John Chambers e colleghi). Gli stessi laboratori noti tra l'altro per lo sviluppo del sistema operativo UNIX o del linguaggio C. R fornisce una valida alternativa Open Source all'ambiente di S, esso è spesso considerato una derivazione di S. L'ambiente R è disponibile come software libero sotto i termini della *GNU General Public License* della *Free Software Foundation* sotto forma di codice sorgente. Si compila e gira su una vasta gamma di piattaforme UNIX e sistemi similari, oltre che su sistemi Windows e MacOS.

5.1 Caratteristiche di R

Lo sbilanciamento verso l'ambiente statistico e matematico non deriva tanto dalla natura del linguaggio, ma dalla disponibilità di grandi raccolte di funzioni statistiche e

[1] http://www.rproject.org/

[2] http://en.wikipedia.org/wiki/S_(programming_language)

Figura 5.1: Logo di R-System

dagli interessi dei ricercatori che lo hanno inventato e lo mantengono. Infatti esso non è solamente un software statistico, in realtà è un ambiente composto da un insieme di funzioni, librerie ed oggetti che possono essere utilizzati per la gestione e l'elaborazione di dati, oltre che per la creazione di grafici ed immagini. Tra le caratteristiche più importanti troviamo:

- un'efficace manipolazione e memorizzazione di dati numerici e testuali grazie alla creazione di *dataFrame*;
- un ben sviluppato semplice ed efficace linguaggio di programmazione che include istruzioni condizionali, loop, funzioni ricorsive definite dall'utente e strumenti di input/output;
- la disponibilità di risorse grafiche per l'analisi e la visualizzazione dei dati;
- un insieme di operatori che consentono di effettuare calcoli su vettori e matrici. Grande semplificazione nel calcolo vettoriale;
- la disponibilità di informazioni e risorse (dall'assistenza alla vasta manualistica scaricabile dalla pagina del progetto);
- un ambiente altamente estensibile attraverso l'installazione di pacchetti (*package*) aggiuntivi. Esistono una grande quantità di moduli (package) distribuiti con licenza GPL e organizzati in un apposito sito chiamato *CRAN*[3] (*Comprehensive R Archive*

[3]http:cran.r-project.org

Network).

R opera tramite linea di comando (figura 5.2) ed eventualmente col supporto di diverse interfacce grafiche (GUI). Tramite la creazione di *script* (file con estension *.r*) si può scrivere codice da mandare in esecuzione direttamente da *shell* utilizzando il comando *source()*. Tra le numerose guide all'introduzione ed all'utilizzo del software si segnala "*Una Guida all'utilizzo dell'Ambiente Statistico R*" [12] e "*Formulario di Statistica con R*" [13].

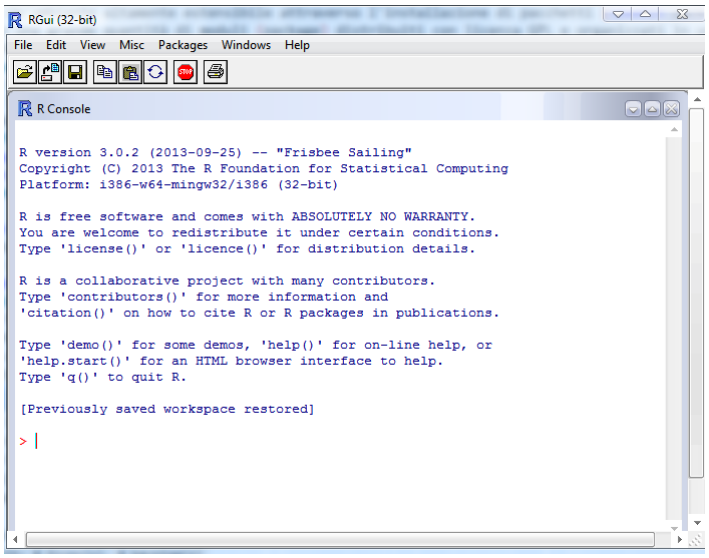

Figura 5.2: R per Windows, versione 3.0.2

5.2 Perchè R?

La scelta di R, come ambiente di sviluppo per questo progetto di tesi, è dovuta al fatto che molti dei problemi che abbiamo incontrato sono gia stati trattati e affrontati da altri. Tra la grande quantità di *package* disponibili sul *CRAN* troviamo:

- moduli per il trattamento delle problematiche di *NLP* (*natural language processing*) e *text mining*. In particolare in questo progetto si utilizza il *package tm*[4]. Esso ci

[4]http://cran.r-project.org/web/packages/tm/index.html

consente di lavorare sul testo dei tweets in maniera intensiva tramite le più comuni operazioni di analisi del testo. Tra queste ricordiamo: la "tokenizzazione" del testo, l'eliminazione di punteggiatura e numeri, il filtraggio del testo, la possibilità di compiere analisi di frequenza delle *word* tramite *Document-term matrix*[5] e molte altre [14];

- moduli per l'interfacciamento alle API di Twitter. Nel nostro lavoro di tesi utilizziamo *streamR*[6] che, ci permette di aprire uno *stream*, per il recupero dei dati, con i server di Twitter [15];

- moduli per la manipolazione e la rappresentazione delle coordinate geografiche. In particolare utilizziamo il *package ggmap*[7] che ci consente l'interfacciamento alle API di Google Maps. Questo ci permette una semplice e intuitiva geo localizzazione dei *tweets* su carta geografica [16].

[5] http://en.wikipedia.org/wiki/Documentterm_matrix
[6] http://cran.r-project.org/web/packages/streamR/
[7] http://cran.r-project.org/web/packages/ggmap/index.html

Capitolo 6

Sentiment Analysis in Twitter: implementazione

In questo capitolo vengono descritte le fasi implementazione e messa in funzione del progetto di tesi; vengono inoltre mostrati i vari *script* (scritti in codice **R**) utilizzati per lo sviluppo delle varie funzionalità. Tutto il codice mostrato nel capitolo è stato eseguito su una singola macchina con processore *dual Core* da 2,1 GHz con RAM da 3 GB.

Il lavoro di progettazione e sviluppo del sistema di *sentiment analysis* si divide in quattro parti principali:

1. filtraggio testuale;
2. addestramento del sistema (*training*);
3. *testing* del sistema;
4. il sistema ed il mondo reale (caso d'uso).

Per la fase di addestramento e di test utilizziamo un *dataset*[1] di circa 600000 *tweets* già etichettati; di questi 2/3 (400000) sono per la fase di *training* e 1/3 (200000) per la fase di *testing*. Ogni riga del *dataset* contiene il testo del *tweet* ed il relativo sentimento espresso (0 per sentimento negativo, 1 per sentimento positivo). Nel listato 6.1 mostriamo lo

[1] http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

script *load.R* che apre il file *dataset*, separa i *tweets* positivi da quelli negativi e infine costruisce quattro *dataframe*:

1. **positiveTrain** formato da 200000 *tweets* positivi (fase di *training*);
2. **negativeTrain** formato da 200000 *tweets* negativi (fase di *training*);
3. **positiveTest** formato da 100000 *tweets* positivi (fase di *testing*);
4. **negativeTrain** formato da 100000 *tweets* negativi (fase di *testing*);

Listato 6.1: Script load.R

```
#
#File: loadDataSet.R
#Author: Damiano Melloncelli

#begin script

file = "SentimentAnalysisDataSet.csv"

linkerFile <- read.csv(file)

data.positive <- data.frame(text= linkerFile[
   linkerFile$sentiment == 1,"textSentiment"])

data.negative <- data.frame(text= linkerFile[
   linkerFile$sentiment == 0,"textSentiment"])

positiveTrain <- data.frame(text=data.positive[1:200000,1])

negativeTrain <- data.frame(text=data.negative[1:200000,1])

positiveTest  <- data.frame(text=data.positive[200001:300000,1])

negativeTest  <- data.frame(text=data.negative[200001:300000,1])
```

```
#end script
```

6.1 Filtraggio

La fase di filtraggio ha come scopo l'eliminazione di tutte quelle componenti testuali non utili alla classificazione, in modo da rendere quest'ultima più efficace ed efficiente. In particolare utilizziamo tecniche di *detection* tramite espressioni regolari per:

- ricerca ed eliminazione di: URL (http...), *hashtag* (#*topic*), **RT** (sigla che indica un *retweet*), **@***user* (menzione di un certo utente *user* nel *tweet*);
- ricerca e sostituzione delle *emoticon*. In particolare quelle che indicano sentimento positivo vengono sostituite dalla stringa "*posmaremo*" (*positive marker emoticon*), mentre quelle che indicano sentimento negativo vengono rimpiazzate dalla stringa "*negmaremo*" (*negative marker emoticon*), vedi tabella 6.1;

Tabella 6.1: Alcune delle Emoticons rilevate

Chiave	Emoticon
posmaremo	(: :) :] [: :-) (-: [-: :-] (; ;) ;] [; ;-) (-; [-; ;-] :-D :D :-p :p (=: ;=D :=) :S @-) XD
negmaremo	:( ): :-( )-: ;( ); :-[ ]-: ;-( )-; :'[:'( )':]: :[:| :/ |: /: :=(:=| :=[xo :| D: O:

- ricerca ed eliminazione delle serie di lettere uguali ripetute più volte, ad esempio la frase "*I looooove youuuu*" diventa *I loove youu*";

- ricerca ed eliminazione di punteggiatura, numeri e parole formate da meno di tre caratteri ;
- eliminazione degli spazzi bianchi ripetuti.

Nel listato 6.2 viene mostrato lo script *filtered.r* che si occupa della fase di filtraggio ricevendo, in input, un *set* di *tweets* non filtrati e restituendo, in output, lo stesso insieme filtrato. Concludendo possiamo osservare un esempio di filtraggio in tabella 6.2.

Listato 6.2: Script filtered.R

```
#
#File: filtered.R
#Author: Damiano Melloncelli

#Begin script

        #** function filterER() -> input: data.frame di tweets
        #**      -> output: data.frame di tweets filtrati
filterER<-function (arg) {

   arg<-apply(arg,2,applyFun)

        #**NOTA: dopo il filtraggio eliminiamo tutti i tweets
        #**che hanno meno di 3 caratterri (poco significativi)
   arg<-data.frame(text=arg[nchar(as.character(arg))>2])
   return(arg)
}

        #** La funzione applyFun esegue il filtraggio su ogni
        #   singolo tweet

applyFun<-function(x){

        #** eliminazione @user
   tmp<-gsub("@.*?(\\s|$)"," ",x);
```

```
     #** eliminazione url (http...)
tmp<-gsub("http.*?$"," ",tmp);

     #** eliminazione #tag
tmp<-gsub("#.*?\\s"," ",tmp);

     #** detection emoticon positive :-), :D ...
tmp<-gsub("(\\(\\:|\\:\\)|\\:\\-\\)|@\\-\\)|\\:O|\\=D|
        \\=\\]|\\=\\)|\\:\\-D|\\:D|\\:\\-p|\\:p|XD|xD|
        \\:\\]|\\;\\]|\\;\\-(\\]|\\))|\\:\\-\\]|\\;\\)
        |\\:\\sD|\\:\\s\\)|D\\:|\\:S|\\:s|\\(\\=|\\^\\
        ^|\\=D)","posmaremo",tmp)

     #** detection emoticon negative )-:, :-/ ...
tmp<-gsub("\\:\\/|\\:\\s\\/|\\:\\'(\\[|\\(|\\{)|\\=\\/
        |\\:\\(|\\)\\:|\\:\\-\\(|\\)\\-\\:|xo|xO|\\:(\
        \(|\\[|\\{|\\|)|\\;(\\(|\\[|\\{\\|)|\\;\\s(\\(
        |\\[|\\{\\|)|\\;\\-(\\(|\\[|\\{\\|)|\\=\\||D\\
        :|\\=(\\(|\\[\\{)","negmaremo ",tmp)

     #** eliminazione serie lettere uguali es.looooove
#tmp<-gsub("(a|b|c|d|e|f|g|h|i|l|m|n|o|p|q|r|s|t|u|v|z
        |j|k|w|x|y){3,30}","\\1\\1",tmp)

     #** eliminazione di numeri e metacaratteri
tmp<-gsub("\\W"," ",tmp)
tmp<-gsub("\\d"," ",tmp)

     #** pulizia testo
tmp<-gsub("(\\s|^)..?\\s"," ",tmp);
tmp<-gsub("(\\s|^).?\\s"," ",tmp);
```

```
    tmp<-gsub("(\\s|^)..?\\s"," ",tmp);
    tmp<-gsub("(\\s|^).?\\s"," ",tmp);

        #** eliminazione spazzi eccedenti
    gsub("\\s{2,30}"," ",tmp)

        #** trasformo le lettere maiuscole in minuscole
    tmp<-tolower(tmp)

    return(tmp)
}

#End script
```

Tabella 6.2: Esempio di filtraggio testuale

Tweet1 originale	#CSexamDate Ooooh my gooood!! Are you sure @lilyRed? tuesday ll be a...a terrible DAY! :-(
Tweet1 filtrato	ooh good are you sure tuesday terrible day negmaremo
Tweet2 originale	It's tooooo funny!!:-D That's crazy! see http://www.youtube.com/watch?v=FydiflH708c
Tweet2 filtrato	too funny posmaremo that crazy see

6.2 Training

Come tutti gli algoritmi di apprendimento (*machine learning*), anche la nostra implementazione del classificatore *Bayesano* necessità di una fase di addestramento, dove il sistema apprende informazioni e nozioni utili per le classificazioni future in modo da

creare una base di conoscenza. Una volta che i *tweets* sono stati filtrati (fase di filtraggio) vengono create due matrici denominate *term document matrix*(*tdm*), una per i termini dei *tweets* positivi, l'altra per i termini dei *tweets* negativi. Una *term document matrix* è una matrice che ha tante colonne quanti sono i documenti e tante righe quanti sono i termini che compongono i documenti. L'elemento E_ij della matrice è settato a 1 se il termine t_i compare nel documento d_j, 0 in caso contrario (vedi listato 6.3). Le *term document matrix* ci permettono di ricavare la frequenza di un termine in un *set* di documenti.

Listato 6.3: Esempio term document matrix

```
#
#File  : tdmExample.R
#Author: Damiano Melloncelli

 #** Prendiamo come esempio i seguenti tweets

   Doc1 = "It is a beautiful day."
   Doc2 = "what day is the test?"
   Doc3 = "my new car is beautiful!"

 #** La term document matrix corrispondente è la seguente

                Docs
      Terms       1 2 3

        a         1 0 0
        beautiful 1 0 1
        car       0 0 1
        day       1 1 0
        is        1 1 1
        it        1 0 0
        my        0 0 1
        new       0 0 1
```

```
    test      0 1 0
    the       0 1 0
    what      0 1 0

#** La matrice ha 3 colonne (una per ogni documento)
#   e 10 righe (una per ogni termine che compare
#   almeno in un documento)
```

Per la creazione delle matrici utilizziamo la funzione *TermDocumentMatrix* del package *tm* che, prende in input un *set* di documenti testuali (*tweets* nel nostro caso) e restituisce in output la relativa matrice *tdm*. La funzione inoltre ci mette a disposizione alcune tecniche di analisi del linguaggio naturale per la riduzione dei termini come: eliminazione delle *stopword*, eliminazione punteggiatura, eliminazione caratteri numerici, *tokenizzazione*, *stemming*, controllo lunghezza termini.

Infine, dalla due *term document matrix*, possiamo ricavare i pesi (frequenza del termine all'interno dei documenti) che ci indicano quanto un termine influenza il sentimento, positivo o negativo, espresso da un *tweet*, o, in altre parole, quanto il sentimento è condizionato dalla presenza del termine. In listato 6.4 mostriamo lo script *frequency.R* che ci permette di ricavare, da una *tdm*, una matrice delle frequenze del tipo $|Termine|Freqenza|$.

Listato 6.4: Script frequency.R

```
#
#File   : frequency.R
#Author: Damiano Melloncelli

#Begin script

   #** function termFrequ() -> input: tdm
   # -> output: data.frame di tipo |termine|frequenza|

termFrequ<-function(matrixTerm)
{
```

```
    matr<<-matrixTerm
    termini<-data.frame (termini=Terms(matr))
    termini[,2]<-apply(termini,1,applyFreq)
    return(termini)
}

    #** La funzione applyFreq esegue la sommatoria di ogni
    #   riga della tdm, in modo da ottenere la frequenza
    #   assoluta di  apparizzione per ogni termine

applyFreq<-function(termine)
{
    val<-as.numeric(tm_tag_score(matr, termine,
       FUN = slam::row_sums))
    return(val)
}

#End script
```

Nota. Nella procedura *apllyFreq()*, in listato 6.4, utilizziamo la funzione *tm_tag_score*() del *package tm* che, individuato un termine all'interno della *tdm*, ne ritorna la frequenza di apparizione nei documenti.

A questo punto abbiamo tutti gli strumenti necessari per avviare la fase di *training*. Nel listato 6.5 mostriamo lo script *train.R* che riassume e ingloba tutte le operazione necessarie per l'addestramento.

Listato 6.5: Script train.R

```
#
#File: train.R
#Author: Damiano Melloncelli

#Begin script

#File: train.R
```

```
#Author: Damiano Melloncelli

#begin script

library(tm)

source("filtered.R")
source("frequenza.R")

    #**Filtraggio
filp<-filterER(positiveTrain[1:200000,1])
filn<-filterER(negativeTrain[1:200000,1])

    #** Creo un tweets corpus positivo
Pcor<-Corpus(VectorSource(positiveTrain[1:200000,1]))

    #** Creo un tweets corpus negativo
Ncor<-Corpus(VectorSource(negativeTrain[1:200000,1]))

    #** Creo la tdm per i termini positivi
learningPositive<-TermDocumentMatrix(Pcor,control =
   list(wordLengths = c(3, 9),removeNumbers=TRUE,
   stopwords=stopwords(kind="en"),removePunctuation
   = list(preserve_intra_word_dashes = TRUE)))

    #** Creo la tdm per i termini negativi
learningNegative<-TermDocumentMatrix(Ncor,control =
   list(wordLengths = c(3, 9),removeNumbers=TRUE,
   stopwords=stopwords(kind="en"),removePunctuation
   = list(preserve_intra_word_dashes = TRUE)))

   #** Creo la matrice delle frequenze per i termini negativi
terminiNeg <- termFrequ(learningNegative)
```

```
   #** Creo la matrice delle frequenze per i termini positivi
terminiPos <- termFrequ(learningPositive)

#end script
```

Nota. Le strutture dati *terminiNeg* e *terminiPos* (vedi ultime righe del listato 6.5) rappresentano la base di conoscenza del nostro classificatore automatico; grazie a loro infatti possiamo determinare quanto un termine influenza il sentimento generale espresso da un *tweet*. In listato 6.6 mostriamo alcuni dati significativi relativi alla fase di addestramento.

Listato 6.6: Script outTrain.R

```
#
#File: outTrain.R
#Author: Damiano Melloncelli

  #** Tdm tweets negativi

> learningNegative (info)

A term-document matrix (58141 terms, 200000 documents)
Non-/sparse entries: 1342971/11626857029

  #** Tdm tweets positivi (info)

> learningPositive
A term-document matrix (62134 terms, 200000 documents)
Non-/sparse entries: 1261500/12425538500

  #** La base di conoscenza è composta da 58141
  #   termini positivi e 62134 termini negativi

> dim(terminiPos)
```

```
[1] 62134     2

> dim(terminiNeg)
[1] 58141     2
>
```

6.2.1 Classificazione

La classificazione, come già detto, avviene tramite l'implementazione di un classificatore Bayesiano. In particolare ogni *tweet* viene scorporato nei propri termini (*token*) e per ognuno di essi si valutano le probabilità che il *tweet*, a cui il termine appartiene, esprima un sentimento positivo e negativo. Riassumendo ogni termine fornisce due dati:

1. probabilità che il *tweet* che lo contiene abbia un orientamento positivo

2. probabilità che il *tweet* che lo contiene abbia un orientamento negativo

Tabella 6.3: Frequenza di apparizione di alcuni termini

Termine	**Frequenza di apparizione**	
	Tw positivi	**Tw** negativi
"love"	12206	4303
"mad"	235	692
"negamaremo"	1973	2034
"posamaremo"	3713	2541
"terrible"	434	77
"sun"	1345	1252
"economy"	20	42

Ad esempio se nella fase di *train* il termine "*economy*" compare in 20 *tweet* positivi e in 42 negativi, un generico *tweet* contenente la parola "*economy*" avrà una maggiore probabilità di risultare negativo durante la classificazione; la tabella 6.3 mostra alcuni termini con relative frequenze di apparizione su i 200000 tweets positivi e su i 200000 tweets negativi utilizzati nella fase di *training*. In listato 6.7 viene mostrato lo script *classification.R* che, mediante l'uso delle matrici di frequenza (da cui ricaviamo tabella 6.3), implementa la funzionalità di classificazione.

Listato 6.7: Script classification.R

```
#
#File: classification.R
#Author: Damiano Melloncelli

#Begin script

   #** function classify() -> input: data.frame di tweets
   #    output: risultati e statistiche di classificazione

classify<-function(tweets)
{
    #** Tempo corrente
   ptm <- proc.time()
   numberT=dim(tweets)[1]
   print(" ...wait for classfication in progress ...")
   matr<-apply(tweets,1,applyClass)

    #** Costruzione output funzione (risultati assoluti)
   retAbs<-data.frame(Positivi=signif(sum(matr[1,1:
      numberT]),5),Negativi=signif(sum(matr[2,1:numberT
      ]),5),Neutrali=signif(sum(matr[3,1:numberT]),5),
      ElapsedTime=signif((proc.time()-ptm)[3],7))

    #** Costruzione output funzione (risultati %)
```

```
    retPer<-data.frame(Positivi=signif((retAbs[1,1]*100)/
        numberT,5),Negativi=signif((retAbs[1,2]*100)/numberT,
        5),Neutrali=signif((retAbs[1,3]*100)/numberT,5),
        ElapsedTime=paste(retAbs[1,4]%/%60,"m",signif(
        retAbs[1,4]%%60,4),"s"))

    r<-rbind(retAbs,retPer)
    dimnames(r)[1]<-list(c("Abs","%"))
    return(r)

}
    #** La funzione applyClass esegue la classificazione su
    #   ogni singolo tweet. Ritorna la polarità del
    #   sentimento espresso:
    # (1,0,0) se il sentimento espreso dal tweet è positivo
    # (0,1,0) se il sentimento espreso dal tweet è negativo
    # (0,0,1) se il sentimento espreso dal tweet è nullo

applyClass<-function(tweet)
{
         #** Tokenizzo il tweet
    toke<-MC_tokenizer(tweet)
    token<-data.frame(toke)
    numberToken<-length(toke)

    if(numberToken>1){
        matri<-apply(token,1,applyToken)

         #** Probabilità che il tweet sia positivo
        probN <- prod (matri[1,1:numberToken])

         #** Probabilità che il tweet sia negativo
        probP <- prod (matri[2,1:numberToken])
```

```
        if(probN>probP)
           ret<-c(0,1,0)

        if(probP>probN)
           ret<- c(1,0,0)

         f(probP==probN)
           ret<- c(0,0,1)
   }
   return(ret)
}
   #** La funzione applyToken ritorna il peso che ha
   #   ogni singolo termine all'interno delle matrici
   #   di frequenza (terminiPos e terminiNeg)

applyToken<-function(tok)
{
   retn=1;
   retp=1;

    #** Prob. che il termine appartenga ad un tweet positivo
   posP<- which(terminiPos[,1]==tok)
   if(length(posP)!=0)
      retp<-as.numeric(terminiPos[posP,2])

    #** Prob. che il termine appartenga ad un tweet negativo
   posN<-which(terminiNeg[,1]==tok)
   if(length(posN)!=0)
      retn<-as.numeric(terminiNeg[posN,2])

   return(c(retn,retp))
}
```

```
#End script
```

6.3 Validation test

Questa fase ha lo scopo di verificare il funzionamento del classificatore valutando accuratezza ed errore. Il test di validazione del sistema avviene tramite l'incremento graduale della dimensione del *training set* e la rispettiva valutazione delle prestazioni ad ogni passo. Ogni *training* è seguito da una fase di test in cui vengono classificati 200000 *tweets* (di cui 100000 positivi e 100000 negativi). Il *training set* di partenza è formato da 2000 *tweets* (di cui 1000 negativi e 1000 positivi), mentre il *training set* di arrivo è formato da 400000 *tweets* (di cui 200000 negativi e 200000 positivi); è indispensabile è che i *tweets* utilizzati per il test siano diversi da quelli utilizzati per l'addestramento. Le misure di accuratezza ed errore per le classi "positivi" e "negativi" vengono calcolate come introdotto in sezione 3.3, mentre i valori di accuratezza ed errore del sistema di classificazione vengono calcolati tramite la media dei risultati sulle singole classi. La tabella 6.4 mostra i risultati dei parametri di *accuracy* ed *error* relativi alle prove eseguite durante il test di validazione; in particolare per ogni *training set* vengono mostrati:

- *accuracy* classi *Positive* e *Negative*, *accuracy* totale del sistema;
- *error* classi *Positive* e *Negative*, *error* totale del sistema;

In figure 6.1 e 6.2 mostriamo i grafici relativi all'andamento dei parametri di *accuracy* ed *error* al variare della dimensione del *training set*. Da questi possiamo notare che i valori di "errore medio" e "accuratezza media" tendono a stabilizzarsi dopo la soglia dei 300000 *tweets*, in particolare osserviamo che alla fine del test di validazione il parametro *error* tende asintoticamente al valore 0.25, mentre il parametro *accuracy* tende asintoticamente al valore 0.75.

Nota. Il *training set* definitivo, utilizzato per il nostro sistema di classificazione, è quindi l'ultimo di tabella 6.4 (200000 *tweets* positivi e 200000 negativi).

Tabella 6.4: Validation test

Tweets	Ac. Pos.	Ac. Neg.	Er. Pos.	Er. Neg.	Accuracy	Error
2000	0,561	0,652	0,439	0,348	0,6065	0,3935
5000	0,573	0,670	0,427	0,330	0,6215	0,3785
10000	0,581	0,735	0,419	0,265	0,6580	0,3420
20000	0,592	0,806	0,408	0,194	0,6990	0,3010
40000	0,609	0,811	0,391	0,189	0,7100	0,2900
80000	0,613	0,830	0,387	0,170	0,7215	0,2785
120000	0,617	0,838	0,383	0,162	0,7275	0,2725
160000	0,623	0,837	0,377	0,163	0,7300	0,2700
200000	0,657	0,821	0,343	0,179	0,7390	0,2610
240000	0,670	0,814	0,330	0,186	0,7420	0,2580
280000	0,684	0,807	0,316	0,193	0,7455	0,2545
320000	0,691	0,803	0,309	0,197	0,7470	0,2530
360000	0,701	0,795	0,299	0,205	0,7480	0,2520
400000	0,719	0,780	0,281	0,220	0,7495	0,2505

6.3.1 Valutazione risultati

Conclusa la fase di *validation test* siamo pronti per la valutazione delle prestazioni del nostro classificatore. In tabella 6.5 vengono mostrati i risultati del *test* di classificazione sul *dataset* contenente i 100000 *tweets* positivi, mentre in tabella 6.6 possiamo osservare i dati relativi ai 100000 *tweets* negativi.

Analizzando le due tabelle osserviamo che il lavoro di classificazione è un compito molto oneroso; per classificare 100000 *tweets* servono circa 4 ore. Notiamo inoltre che abbiamo migliori risultati nell'esecuzione del secondo test (*tweets* negativi), ciò implica un piccolo sbilanciamento verso un lato della classificazione. Ironicamente potremmo definire il nostro classificatore "leggermente pessimista".

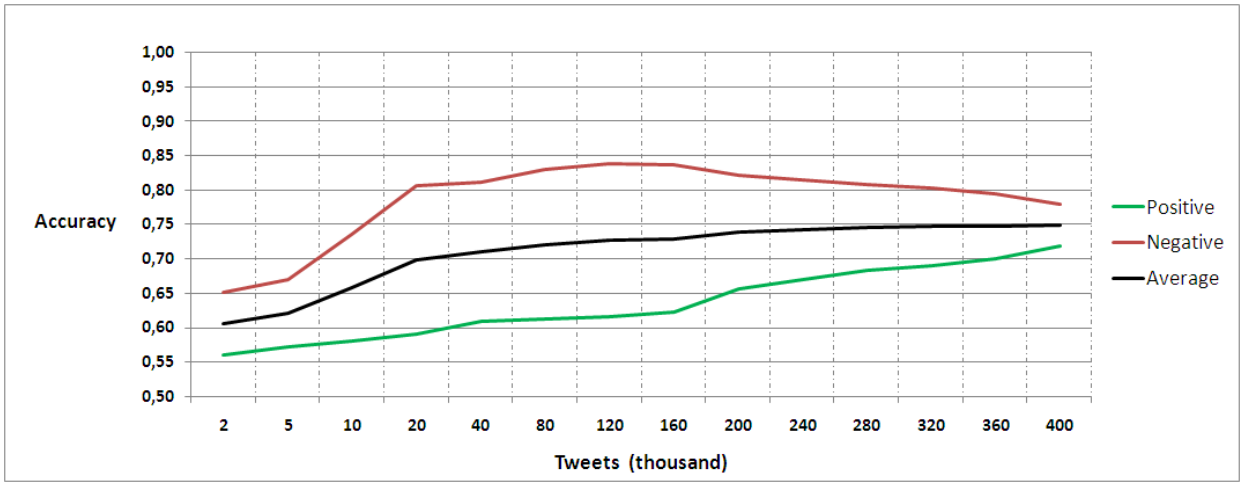

Figura 6.1: Validation: andamento accuracy

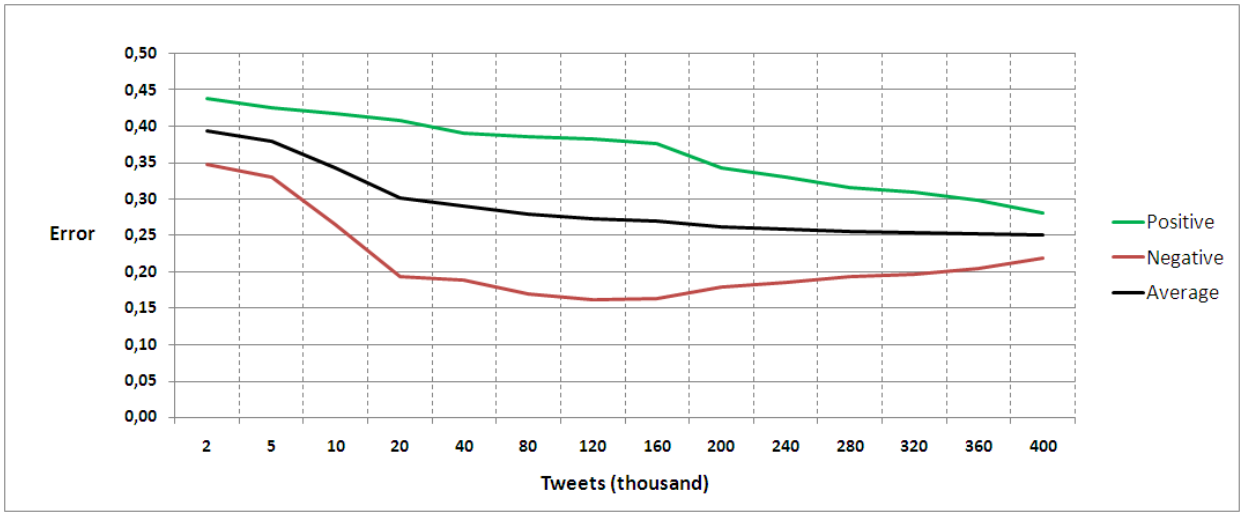

Figura 6.2: Validation: andamento error

Fondendo i risultati dei due test (vedi tabella 6.7) otteniamo i seguenti risultati:

- nel 75% dei casi il sistema rileva in modo esatto il sentimento espresso;
- nel 24% dei casi il sistema rileva un sentimento contrario rispetto a quello espresso
- nel 1% dei casi il sistema non è in grado di valutare il sentimento espresso

Tabella 6.5: Test classificazione tweets positivi

	Positivi	**Negativi**	**Neutrali**	**Tempo di esecuzione** (s)
Abs.	71910	27078	1012	12954
%	71,91 1	27,08	1,01	

Tabella 6.6: Test classificazione tweets negativi

	Positivi	**Negativi**	**Neutrali**	**Tempo di esecuzione** (s)
Abs.	21258	78008	734	12915
%	21.26	78.01	0.73	

Tabella 6.7: Prestazioni classificatore

	Accuratezza	Errore	Non classificabile
%	74,95	24,18	0,87

6.4 Il sistema ed il mondo reale

Abbandoniamo il mondo ovattato e irreale della fase di test dove, le informazioni (*tweets*) che ci servivano erano contenute in un *dataset* pre-costruito e pre-classificato, e proviamo a proiettare il nostro sistema nel mondo reale dove, le informazioni che vogliamo provare a classificare ci vengono date direttamente dai server di Twitter. Affrontiamo in questa sezione le seguenti problematiche:

- connessione ai server di Twitter e *tweets mining*;
- memorizzazione informazioni ed eliminazione *tweets* ripetuti;
- geo-localizzazione *tweets* (dove è possibile).

6.4.1 Tweets mining

Per il recupero dei *tweets* utilizziamo il package *streamR*; esso, implementando le funzionalita delle *streaming API* di Twitter, fornisce gli strumenti per la creazione di connessioni ai server per il *download* dei dati. Le *streaming API* di Twitter richiedono autenticazione tramite il protocollo OAuth, implementato dal package *ROAuth* di **R**. L'autenticazione avviene tramite le credenziali di accesso a Twitter; è permessa una sola connessione di *streaming* per utente autenticato. Una volta autenticati si ottiene un *token* che non ha scadenza e che deve essere utilizzato per tutte le richieste di connessione future. La funzione *filterStream* del package *streamR* ci permette di aprire una connessione *stream* per il recupuro dei *tweets*, essa ci permette di impostare *key words* e *box* geografiche per la ricerca. In listato 6.8 mostriamo lo script *tweetMining.R* che descrive tutti i passaggi necessari alla richiesta del *token* e all'autenticazione, infine esegue una richiesta di *streaming* tramite l'invocazione del metodo *filterStream*.

Listato 6.8: Script tweetMining.R

```
#
#File: tweetMining.R
#Author: Damiano Melloncelli

#Begin script

library(twitteR)
library(ROAuth)

 #** Richiesta autenticazione

reqURL <- "https://api.twitter.com/oauth/request_token"

accessURL <- "http://api.twitter.com/oauth/access_token"

authURL <- "http://api.twitter.com/oauth/authorize"

 #** Key e Secret si richiedono su https://dev.twitter.com/
```

```
 #   dopo essersi registrati come sviluppatore

consumerKey <- YOUR_KEY
consumerSecret <- YOUR_SECRET

my_oauth <- OAuthFactory$new(consumerKey=consumerKey,
   consumerSecret=consumerSecret,requestURL=reqURL,
   accessURL=accessURL, authURL=authURL)

 #** Richiesta certificato per autenticazione
download.file(url="http://curl.haxx.se/ca/cacert.pem",
   destfile="cacert.pem")

my_oauth$handshake(cainfo="cacert.pem")

 #** Token registrato
registerTwitterOAuth(my_oauth)

 #**Esempio di chiamata funzione filterStream()
filterStream( file="prova.json",track="Bologna", timeout=600,
   oauth=my_oauth )

 #** I tweets recuperati, contenenti la keyword "Bologna",
 #   vengono salvati nel file prova.json. Lo streaming ha
 #   una durata di 600 secondi e Utilizza il token
 #   precedentemnete registrato per l'autenticazione

#End script
```

6.4.2 File *json* ed estrazione testo

Concluso lo *stream* della funzione *filterStream()* i dati recuperati vengono salvati in un file *.json*[2]. La funzione *parseTweet()* del package *streamR* ci consente di eseguire una traduzione del file *.json* consentendoci la manipolazione dei dati scaricati. Ricordiamo che un singolo *tweet* recuperato tramite *streaming API* comprende, oltre al campo testuale, molte altre informazioni; le più significative per noi sono: *id_tweet*, *latitude e longitude*. Le coordinate geografiche non sono sempre presenti, dipende infatti dal dispositivo da cui è stato inviato il messaggio e dalle *policy* di utilizzo del servizio; è quindi necessario dividere i *tweets* geo-localizzati da quelli non. Un altro problema è la gestione dei "doppioni": le *streaming API* di Twitter restituiscono molti *tweets* ripetuti. Tramite la verifica del campo *id_tweet* (identificatore univoco per ogni *tweet)* è possibile eliminare tutti i messaggi ripetuti. In listato 6.9 possiamo osservare lo script *estraiParse.R* che mostra l'implementazione delle funzionalità sopra descritte.

Listato 6.9: Script estraiParse.R

```
#
#File: parse.R
#Author: Damiano Melloncelli

#Begin script

  #**La funzione miningData() prende in input un file.json
miningData<-function(filename)
{
   ptm <- proc.time()

  #**Parsing del file .json
   tweet<-parseTweets(filename)

  #**Numero di tweets recuperati
   nt<-dim(tweet)[1]
```

[2]http://it.wikipedia.org/wiki/JSON

```
#**Leggo il file IDTWEET con gli id dei tweet esaminati
#   fino a questo momento
 pt      <- read.csv("IDTWEET.csv")
 id.set <<- data.frame(id=as.numeric(pt[,2]))

 tweet<-apply(tweet,1,applyHash)

#**Indici dei tweets non ripetuti (da memorizzare)
 num<-which(tweet[1,]!=-1)
 num1=c(1,3,4)

#**Creo un dataSet escludendo il testo
 data.info<<-data.frame(tweet[num1,num])

#**Indici dei tweets che possiedono cordinate geografiche
 geonum<-which(is.na(data.info[3,])==FALSE & is.na(
    data.info[2,])==FALSE)

#**Indici dei tweets che non possiedono cordinate geografiche
 num<-setdiff(num,geonum)

 data.text<-c(tweet[2,num])
 data.textGeo<-c(tweet[2,geonum])

#** Creazione strutture dati output funzione
#   nogeoTweet-> testo tweets non geo localizzati
#   geoTweet  -> testo tweets non geo localizzati
#   geoInfo   -> coordinate tweets geo localizzati
 nogeoTweet<<-mininText(data.text)
 geoTweet  <<-mininText(data.textGeo)
 geoInfo   <<-data.frame(data.info[,geonum])
```

```
   nt<- nt-length(data.text)-length(data.textGeo)
   print(paste(as.character(nt) ," tweet rimosi perche doppi"))

  #** Aggiorno il file IDTWEET con gli id dei tweet estratti
  #   dal file .json
   write.csv(id.set,"IDTWEET.csv")
   print(proc.time()-ptm)

}

  #** La funzione applyHash riceve un payload di un  tweet,
  #   controlla se l'ID_TWEET non compare gia nel set
  #   estrato dal file IDTWEET. Ritorna in caso affermativo:
  #   (-1,text,lat,lon), in caso negativo (ID_TWEET,text,lat,lon)
applyHash<-function(tw)
{
   if(is.element(tw[26],id.set[,1])==FALSE)
   {
        ret <- c(tw[26],tw[1],as.numeric(tw[38]),
           as.numeric(tw[37]))
        dat<-data.frame(id=as.numeric(tw[26]))

  #** Aggiorno il dataSet degli ID_TWEET con il nuovo
  #   IDTWEET recuperato
        id.set<<-(rbind(id.set,dat))
   }
   else
        ret <- c(-1,tw[1],as.numeric(tw[38]),
           as.numeric(tw[37]))

   return(ret)
}
```

```
  #** La funzione mininText riceve un'array di tweets, ritorna
  #   un dataFrame di tweets; è una funzione di utilità che
  #   garantisce l'efficace memorizzazione dei caratteri testuali
  #   per le successive fasi di filtraggio e classificazione.
mininText <- function(data)
{
   n<-length(data)
   string<-data[1]
   ret<-data.frame(text=string)
   for(i in 2 : n){
        string<-data[i]
        dat<-data.frame(text=string)
        ret<-rbind(ret,dat)
   }
   return(ret)
}

#End script
```

Eseguita la funzione *miningData()* siamo pronti per il processo di classificazione dei *tweets*, contenuti nei *dataset geoTweet* e *nogeoTweet*, come visto nella fase di *test* (vedi sezione 6.3), procedendo con il filtraggio dei dati e la successiva classificazione.

6.4.3 Geo-localizzazione

Oltre alla determinazione dell'orientamento (sentimento), risulta interessante anche localizzare geograficamente il *tweet* per sapere quindi dove il "sentimento" è stato espresso. Alla funzione *classify*, vista in listato 6.7, viene apportata qualche lieve modifica per permettere, durante la classificazione del *dataset geoTweet*, la costruzione di una matrice del tipo

$$|Latitudine|Longitudine|Sentimento|$$

Grazie a quest'ultima possiamo localizzare i *tweets* su una mappa geografica distinguendo i positivi dai negativi.

Il package *ggmap* ci mette a disposizione, in modo semplice e intuitivo, i principali metodi forniti dalle *Google Map Api*[3]; in particolare utilizziamo la funzione *get_googlemap()* che ci permette di richiedere una carta geografica (stile *Maps* di Google) con la possibilità di variare località e zoom. Ottenuta la mappa, andremo a "marcarla" con i punti ricavati dalla matrice descritta sopra, in modo da visualizzare la locazione dei *tweets*. In listato 6.10 mostriamo lo script *geoTweets.R* contenente la funzione *drawMap* che esegue una richiesta di mappa e ne permette la visualizzazione marcando i *tweets*.

Listato 6.10: Script geoTweets.R

```
#
#File: geoTweets.R
#Author: Damiano Melloncelli

#Begin script

    #** function drawMap() ->input: matrice del tipo
    #   |lat|lon|sent(0/1)|; output -> mappa
    #   geografica con tweets localizzati
drawMap<-function(cord){

    #** Marco sent==0 come sentimento negativo
    #** sent==0 come sentimento negativo
   cordi<<-data.frame(lat=cord$lat, lon=cord$lon)
   v<-vector( )
   for(i in 1: dim(cord)[1]){
        if(cord[i,3]==0) v[i]="Negative"
        if(cord[i,3]==1) v[i]="Positive"
   }
   cordi<-cbind(cordi,v)
   names(cordi) <- c("lat", "lon","Sentiment")

    #** Imposto il punto centrale della mia mappa
```

[3]https://developers.google.com/maps/

```
    center<-as.numeric( geocode("Via Zamboni, Bologna"))

     #** Richiesta mappa centrata in "Bologna". Vista Italia
     #   centro-settentrionale (zoom=7)
    map<-get_googlemap(center="bologna",size=c(640,644),
        fullscreen="TRUE",zoom=7)

     #** Visualizzo l'oggetto map come un immagine, tramite
     #   la funzione geom_point() marco i punti dati dalle
     #   cordinate passate in input
    ggmap(map,leggend="top") + geom_point(aes(y = lat,
        x = lon,colour=Sentiment),data = cordi, size=2)

#End script

}
```

Variando il parametro *zoom* nella funzione *get_googlemap()* aumentiamo o diminuiamo il dettaglio mappa. Ne mostriamo un esempio visualizzando la marcatura di 60 *tweets* catturati con la funzione *filterStream()* (impostando come *box* geografica la zona di Bologna). In figura 6.3 è stato impostato il parametro *zoom* a 8, in figura 6.4 a 11 e in figura 6.5 a 15.

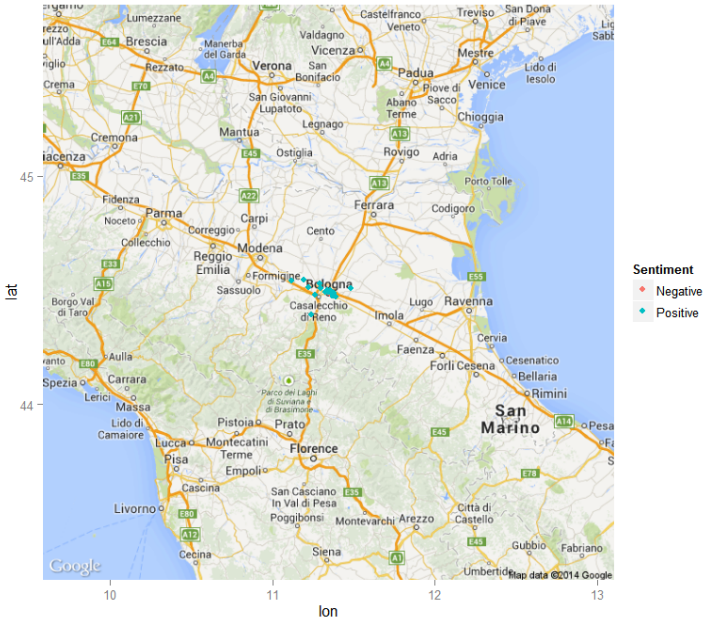

Figura 6.3: Esempio mappa con zoom = 8

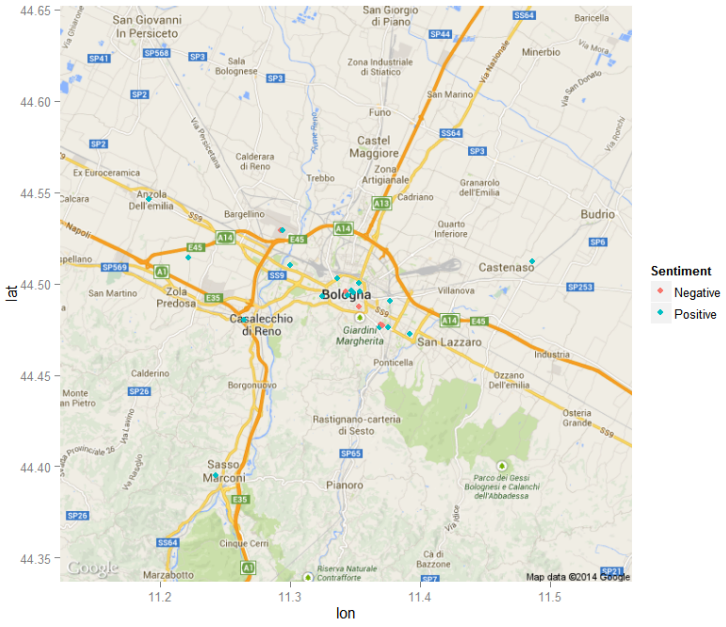

Figura 6.4: Esempio mappa con zoom = 11

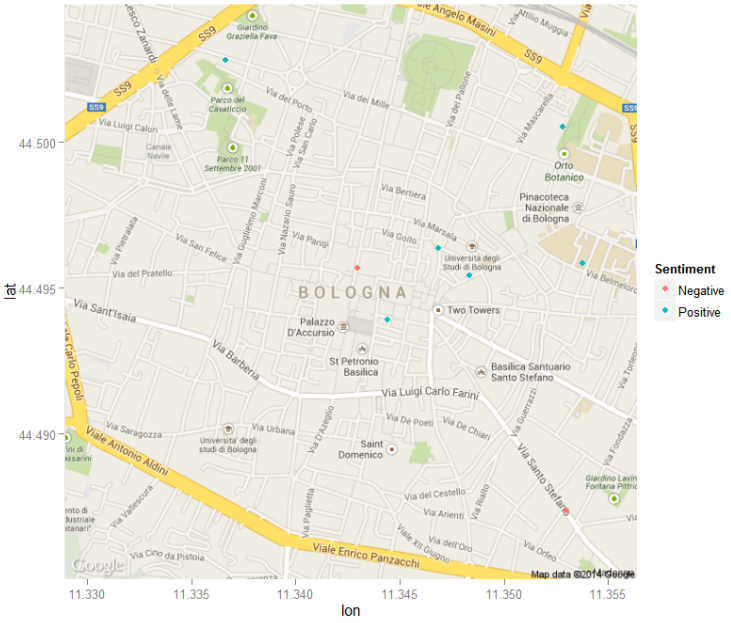

Figura 6.5: Esempio mappa con zoom = 15

I vantaggi di una localizzazione così precisa sono enormi, infatti con questo tipo di analisi è possibile localizzare il sentimento anche per piccole aree geografiche; ad esempio potremo capire cosa pensa la gente di un determinato quartiere o addirittura di una determinata via, degli amministratori della propria città. Purtroppo però, come già detto, solo una percentuale molto bassa di *tweets* comprende nel proprio *payload* le informazioni necessarie alla localizzazione (latitudine e longitudine).

Capitolo 7

Caso studio: Obama's Popularity

In questo capitolo si propone la descrizione di un piccolo caso studio che ci ha permesso di testare le funzionalità del nostro classificatore applicandolo ad un problema reale. L'idea è quella di sondare e rilevare l'opinione generale di un certo numero di utenti del *social network* Twitter relativamente ad un certo argomento; in particolare ci siamo posti questo quesito: cosa pensano gli utenti di Twitter del presisdente Barack Obama? Modellando la domanda rispetto al progetto *Sentiment Analysis in Twitter* potremo ottenere una riformulazione del tipo: l'opinione generale espressa dagli utenti di Twitter relativamente al presidente Obama è positiva o negativa?

7.1 Un Benchmark per il caso studio

In campo economico, una plausibile definizione del termine *Benchmark* è la seguente:

"*indicatore, misura, parametro di riferimento in base al quale un'azienda valuta le proprie prestazioni relativamente a prodotti, servizi, processi aziendali*[1]"

Pur non essendo un azienda, anche il nostro classificatore necessita di un *set* di dati di riferimento (*Benchmark*) per eseguire un confronto, allo scopo di valutare la correttezza e le prestazioni del sistema di classificazione. Trattandosi di un personaggio politico, lo strumento di confronto più potente è il sondaggio politico. Fortunatamente le indagini

[1] http://www.formazionearte.it/GLOSSARIO%20DELLA%20COMUNICAZIONE.doc

statistiche, per la rilevazione della popolarità del presidente degli Stati Uniti, sono molto frequenti, quasi giornaliere. Le informazioni sono reperibili abbastanza facilmente in rete presso siti web di quotidiani e di attualità politica, un di questi è RCP[2]. Real Clear Politics è uno dei principali siti web che raccoglie indagini, attualità, sondaggi e informazioni in genere relativamente al mondo politico statunitense. Il 17 Febbraio 2014,

Tabella 7.1: RCP: Obama Job Approval

Poll	**Date**	**Sample**	**Approve %**	**Disapprove %**
RCP Average	2/4 - 2/16		43	52.8
Gallup	2/14 - 2/16	1500	41	53
Rasmussen	2/14 - 2/16	1500	48	50
FOX News	2/9 - 2/11	1006	42	53
The Economist/YouGov	2/8 - 2/10	716	42	56
McClatchy/Marist	2/4 - 2/9	970	42	52

RCP pubblica *President Obama Job Approval*[3], un articolo dove vengono aggregati i risultati di diversi sondaggi di opinione (**Gallup**[4], **Rasmussen Reports**[5], **FOX News**[6], **The Economist/YouGov**[7], **McClatchy/Marist**[8]) relativi all'argomento "*Obama job approval*" eseguiti sul territorio statunitense tra il 4 e il 16 Febbario 2014. In tabella 7.1 mostriamo un estratto dei dai dell'articolo *Obama job approval* in cui si evince che il 43% degli intervistati ha un opinione generale positiva relativamente al lavoro svolto

[2]http://www.realclearpolitics.com/

[3]http://www.realclearpolitics.com/epolls/other/president_obama_job_approval-1044.html

[4]http://www.gallup.com/poll/113980/Gallup-Daily-Obama-Job-Approval.aspx

[5]http://www.rasmussenreports.com/public_content/politics/obama_administration/obama_approval_index_history

[6]http://www.foxnews.com/politics/interactive/2014/02/12/fox-news-poll-voters-think-worst-yet-to-come-on-economy-disapprove-obama/

[7]http://d25d2506sfb94s.cloudfront.net/cumulus_uploads/document/w94t9k1idp/ trackingreport.pdf

[8]https://s3.amazonaws.com/s3.documentcloud.org/documents/1016656/mcclatchy-marist-poll-obama-congress-nature-of.pdf

dal presidente Obama, mentre il 52,8 % ha invece un opinione generale negativa. Questi dati rappresentano il *Benchmark* per il nostro caso studio.

7.2 Sviluppo

Utilizzando le *streaming API* di Twitter (come mostrato in capitolo 6) sono stati scaricati *tweets* scritti in lingua inglese, contenenti una tra le seguenti *keywords*: *"Obama", "obama", "OBAMA"*. Il periodo di *mining* è iniziato alle ore 00:01 del giorno 01-02-2014 e si è concluso alle ore 23:59 del giorno 05-02-2014. Queste 120 ore di *streaming* hanno portato alla creazione di 5 file (uno per giorno), formando un *dataset* globale contente 555321 *tweets*; da ogni file sono stati creati due *dataset* (vedi tabella 7.2)

- *geoTweet*: formato dai i *tweets* contenti informazioni relative alla geo-localizzazione;
- *nogeoTweet*: formato dai i *tweets* non contenti informazioni relative alla geo-localizzazione;

Dalla tabella 7.2 notiamo che il numero di *tweets*, contenenti informazioni utili relativamente alla geo-localizzazione, è una piccolissima percentuale del totale (nemmeno l'un percento). Il processo prevede la classificazione dei *dataset* di ogni singolo giorno al fine di determinare l'andamento del *sentiment* nel tempo. In tabella 7.3 mostriamo i risultati assoluti e percentuali della classificazione avvenuta sui *set* dei 5 giorni, mentre in figura 7.1 possiamo osservare l'andamento del *sentiment* espresso nel tempo.

I *tweet* in categoria "**Non class.**" sono quelli a cui il classificatore non è stato in grado di attribuire un "*sentiment*", questo può avvenire per uno dei due possibili motivi:

1. il campo testuale del *tweet* non contiene termini presenti nella nostra base di conoscenza (cioè l'insieme dei termini incontrati durante la fase di *training* per cui esistono valori di probabilità);
2. analizzando i termini presenti nel campo testuale del *tweet*, il classificatore giunge alla conclusione che la probabilità di appartenere alla classe dei negativi è uguale alla probabilità di appartenere alla classe dei positivi (neutralità).

Tabella 7.2: Creazione dei dataset geoTweet e nogeoTweet

File	nogeoTweet (dimensione)	geoTweet (dimensione)
Obama_01-feb	113725	1131
Obama_02-feb	112217	1231
Obama_03-feb	106326	779
Obama_04-feb	104380	1236
Obama_05-feb	113340	956
Totale	**549988**	**5333**

Tabella 7.3: Risultati classificazione

	Assoluti			**Percentuali**		
	Positive	Negative	Non class.	% Positivi	% Negative	% Non class.
Day1	53372	46598	12644	47,39	41,38	11,23
Day2	53400	50341	9705	47,07	44,37	8,55
Day3	48232	46953	10944	45,45	44,24	10,31
Day4	47026	45998	11031	45,19	44,21	10,60
Day5	55445	47132	10995	48,82	41,50	9,68
	Average			**46,78**	**43,14**	**10,08**

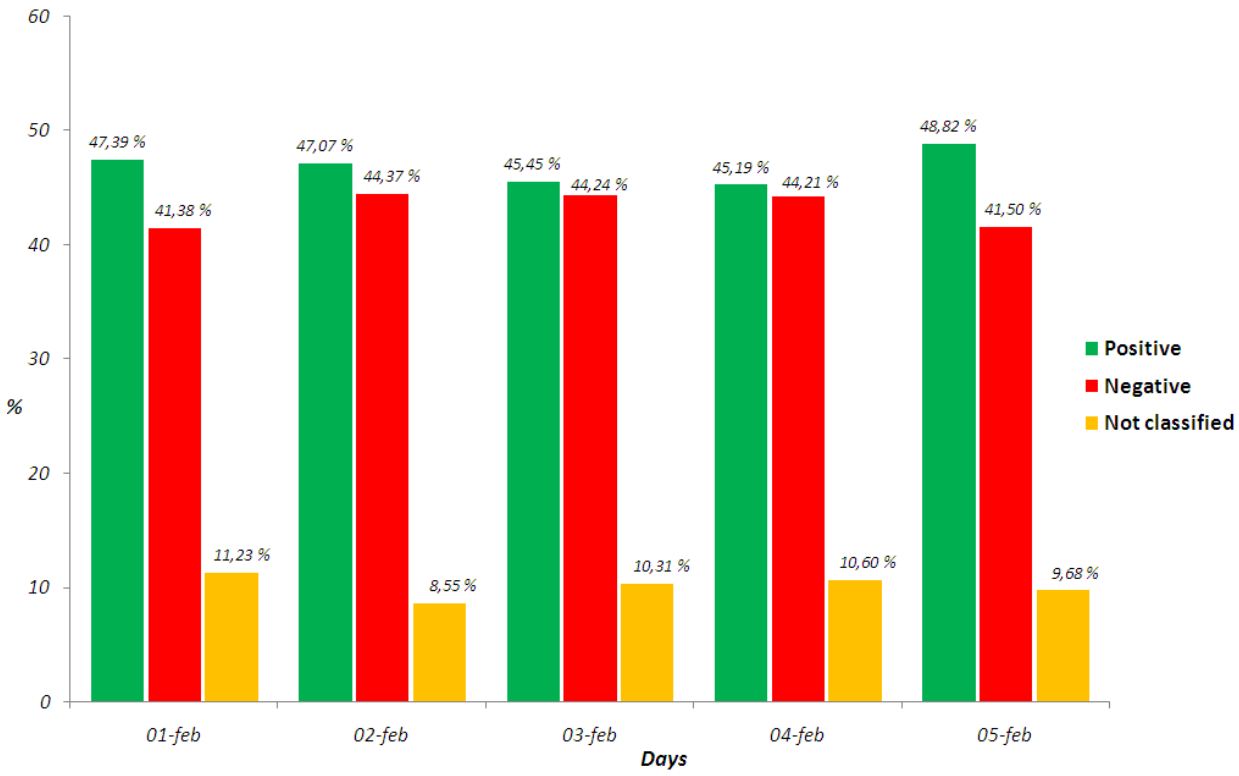

Figura 7.1: Analisi temporale del sentiment

Volendo riassumere i risultati ottenuti, abbiamo che:

- il 46.78% dei *tweet* analizzati esprimono un *sentiment* generalmente positivo;
- il 43.14% dei *tweet* analizzati esprimono un *sentiment* generalmente negativo;
- il 10.08% dei *tweet* analizzati non sono classificabili, cioè non esprimono ne un *sentiment* negativo, ne un *sentiment* positivo.

7.3 Geo-localizzazione

I *tweets* geo-localizzati rappresentano una piccolissima parte del nostro campione (vedi tabella 7.2); per questo motivo un indagine accurata sulla distribuzione spaziale del sentimento non ha molto senso. In presenza di una frequenza maggiore dei *tweets* geo-localizzati le applicazioni potevano molteplici. Mostriamo comunque in figura 7.2 la localizzazione del sentimento relativo all'intero territorio statunitense, in figura 7.3 la localizzazione del sentimento relativo ad alcuni stati del nord-est e in figura 7.4 la localizzazione del sentimento relativo alla città di New York.

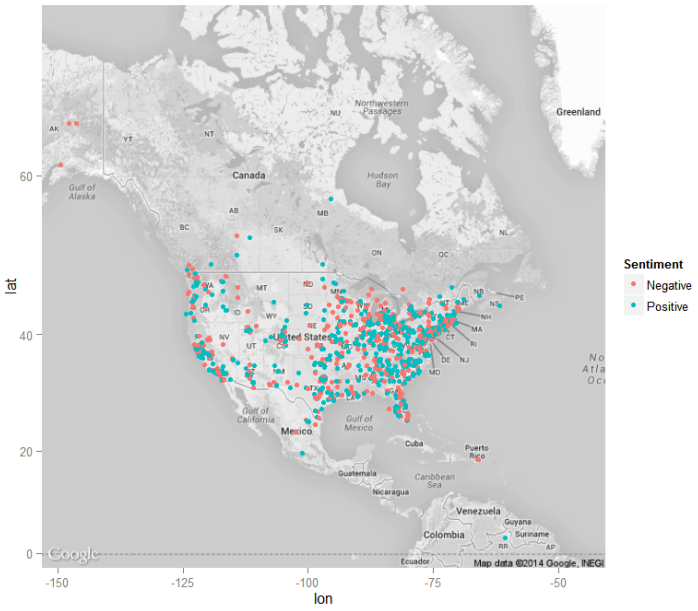

Figura 7.2: Mappa USA con sentiment analysis

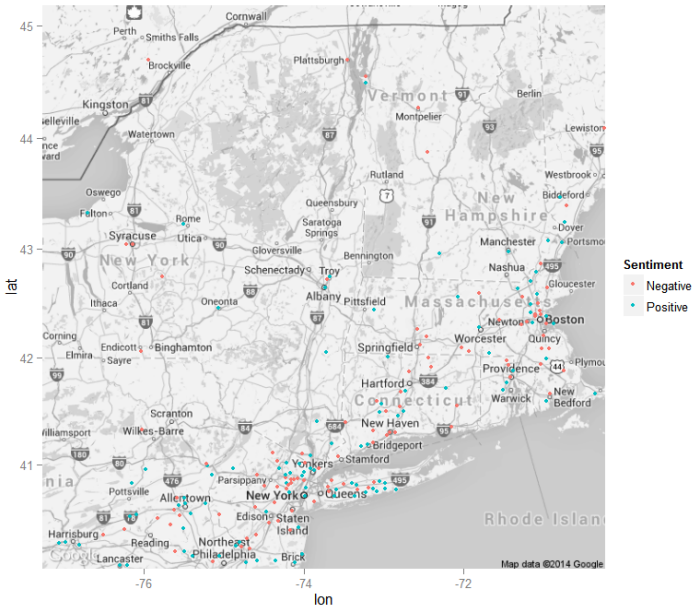

Figura 7.3: Mappa stati nord-est USA con sentiment analysis

Figura 7.4: Mappa New York City con sentiment analysis

7.4 Confronto risultati

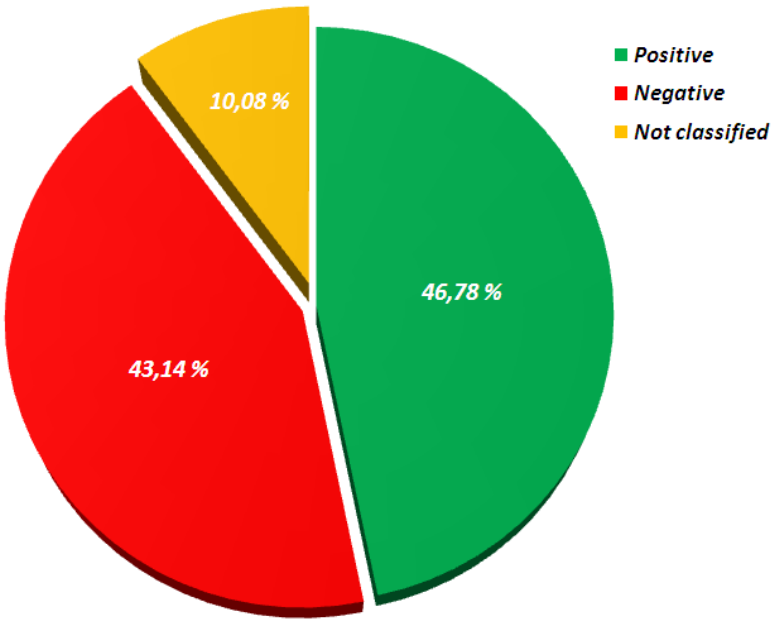

Figura 7.5: Caso studio: Obama's Popularity

I risultati della classificazione (vedi figura 7.5) non rispecchiano evidentemente quelli forniti da RCP (vedi figura 7.6), evidenziamo in particolare un margine (differenza) di 9 punti percentuali per quanto riguarda la classe *Positive* e 4 punti percentuali relativamente alla classe *Negative*; i possibili fattori che vanno a determinare questo margine potrebbero riguardare:

Campione analizzato. I sondaggi proposti da RPC si basano su un attenta selezione del campione statistico, in particolare questi tipi di indagine vanno a sondare un campione equi distribuito in termini di sesso, età, orientamento politico, razza. Noi purtroppo non possediamo nessuna informazione di questo tipo relativamente al nostro campione; è da segnalare anche la notevole differenza numerica fra i due campioni (circa 6000 campionature in RCP contro il mezzo milione del nostro *Obama's Popularity*)

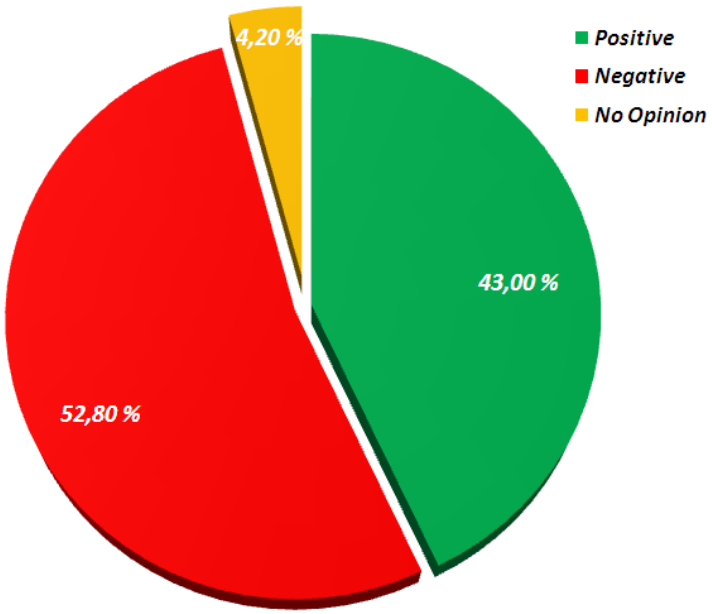

Figura 7.6: RCP: President Obama Job Approval

Not classified. Il 10% del nostro campione (circa 50000 *tweets*) risulta non classificabile; è possibile che classificando questa parte del campione in modo corretto il risultato finale venga ribaltato, avvicinandosi di molto ai dati forniti da RCP.

Periodo di rilevazione. Il periodo di rilevazione non è il medesimo nelle due indagini: RCP propone una serie di risultati ottenuti da rilevazioni avvenute nel periodo che va dal 2 febbraio 2014 al 16 febbraio 2014, mentre il caso studio *Obama's Popularity* utilizza dati rilevati nel periodo che va dal 1 febbraio 2014 al 5 febbraio 2014.

Ultimo, ma non meno importante, aspetto da considerare è il fattore psicologico; l'intervistato nei sondaggi è invitato ad esprimere un opinione relativamente ad un certo argomento in un determinato momento della giornata: parliamo quindi di una rilevazione precisa e puntuale. L'utente del *social network* quando *twitta* il proprio messaggio lo fà in piena libertà, magari senza l'intenzione di esprimere un opinione, egli infatti sa che ciò che scrive non sarà sottoposto ad una valutazione per determinare l'orientamento della propria opinione riguardo ad un certo fatto (cosa che succede in un sondaggio). Questo aspetto non è certamente da sottovalutare e rappresenta secondo noi uno dei punti di

forza della *sentiment analysis*, infatti la libertà di espressione fornita dai *social network*, slegata da tutti i vincoli legati ai formalismi di un sondaggio o di un'indagine statistica, rappresenta secondo noi la vera e sincera forma di opinione.

Capitolo 8

Conclusioni

Il lavoro di tesi ha presentato aspetti e problematiche relative al settori di studio della *sentiment analysis*. Partendo con un introduzione teorica, sulle metodologie e le tecniche utilizzate, nel campo della *sentiment analysis* e della *text categorization*, si è giunti alla progettazione del sistema *Sentiment Analysis in Twitter* che utilizza la classificazione testuale (della sola lingua inglese) per determinare l'orientamento (il *sentiment*) dei messaggi postati sul *social network* Twitter.

Per lo sviluppo dell'applicativo si è utilizzato il *tool* **R Project** in quanto fornisce un ambiente *open source* espandibile tramite l'integrazione di un gran numero di *package* dalle più disparate funzionalità (Twitter API, *natural language programming*, Google Maps API). Il sistema di classificazione, preso in input un *tweet*, restituisce in output la classe a cui il *tweet* appartiene, scelta tra: classe *positive*, classe *negative*, *not classified*. *Sentiment Analysis in Twitter* prevede una fase di pre-classificazione, denominata "filtraggio", in cui il testo dei *tweet* viene "pulito" da tutto ciò che non è utile ai fini della classificazione; in particolare in questa fase viene eseguita la *detection* e il *tagging* delle emoticons, che rappresentano un importante indicatore del *sentiment* espresso.

Il sistema è stato addestrato con una tecnica di *cross validation* incrementale, cioè il *training set*, inizialmente molto piccolo, viene incrementato gradualmente e ad ogni passo vengono valutati i risultati della classificazione su un *test set*. Questa procedura continua fino a che non vengono raggiunte le prestazioni ottimali del sistema o fino a quando non viene esaurito il *training set* che si ha a disposizione. Durante questa fase

il sistema ha raggiunto, come sua massima e migliore prestazione, un *rating* (accuracy) di classificazione pari al 75%, in pratica significa che nel 75% dei casi il sistema è in grado di classificare il *tweet* nella relativa corretta classe di appartenenza. C'è da dire che risultati migliori sono stati ottenuti nei lavori in [7] e [11], usando diverse tecniche di estrazione dei termini e diversi metodi di classificazione, arrivando ad un *accuracy* con valori compresi tra 80% e 82%.

L'ultima parte della tesi è dedicata ad un caso studio in cui si è cercato di determinare l'opinione degli utenti di Twitter relativamente al presidente degli USA Barack Obama. I risultati ottenuti, anche se diversi da quelli forniti da sondaggi e indagini statistiche, sono abbastanza soddisfacenti, non tanto per il risultato in se, ma in quanto dimostrano quanto sia facile raggiungere un vastissimo campione utilizzando metodi di *sentiment analysis* su *social network*, cosa praticamente irrealizzabile tramite un sondaggio o una classica indagine statistica . Dopo la classificazione si è cercato di localizzare geograficamente i messaggi postati, in modo da compiere un analisi spaziale del *sentiment*, questo però non è stato possibile a causa del bassissimo numero dei messaggi contenti informazioni utili alla geo-localizzazione (latitudine e longitudine); fatto dovuto alle limitazioni date dalla diversità dei dispositivi usati (*smartphones*, PC, ecc...) e dalle *policy* di utilizzo di Twitter.

Possibili sviluppi futuri di questo lavoro di tesi includono:

- impiego e test di diversi tipi di classificatori, oltre al *Naive Bayes* utilizzato nella nostra implementazione, al fine di aumentare le prestazioni del sistema di classificazione;
- adozione di un sistema di pre-classificazione al fine di discriminare i messaggi oggettivi, che non presentano sentimenti o opinioni ma esprimono solamente fatti, dai messaggi soggettivi che realmente esprimono un orientamento personale (*sentiment*);
- utilizzo di tecniche basate sulla *sentiment consistency* per cercare di rilevare ironia e sarcasmo all'interno dei *tweets*.

Concludendo ritorniamo alle motivazioni di questo lavoro di tesi, sottolineando che l'utilizzo di tecniche, di *sentiment analysis* su *social network*, consente la rilevazione

autentica del libero pensiero, espressione o opinione. La gente utilizza liberamente questi servizi perché ha voglia di esternare le proprie emozioni e non perché qualcuno gli chiede di farlo. Ricordiamo una delle frasi simbolo di un altro celebre *social network* (Facebook), che sintetizza al meglio il concetto sopra descritto: "*What's on your mind*?". Un breve appunto sulla questione etica, in [18] l'autore scrive:

"*Siamo tutti "pollicini" digitali e molte delle briciole che lasciamo dietro di noi sono espresse in "linguaggio naturale.*"

Sono proprio queste "briciole che vengono utilizzate nella *sentiment analysis*; sempre in [18] l'autore si pone la seguente domanda: "Stanno spiando le nostre emozioni?" Nell'era in cui ogni cosa, fatto o avvenimento che ci interessa viene condiviso, proprio per far sapere agli altri che ci interessa, potremmo rispondere alla domanda sopra citata con un'affermazione del tipo:

"*Si stanno spiando le nostre emozioni,*
ma solo perché noi lo vogliamo".

Bibliografia

[1] B. Liu and L. Zhang. *A survey of opinion mining and sentiment analysis.* In "Mining Text Data", pages 415-463, Springer, 2012.

[2] N. Jindal and B. Liu. *Opinion Spam and Analysis.* In Proceedings of the International Conference on Web Search and Data Mining, pages 219-230, 2008.

[3] B. Pang and L. Lee. *Thumbs up? Sentiment Classification using Machine Learning.* In Proceeding of the ACL-02 conference on Empirical methods in natural language processing, pages 79-86, 2002.

[4] B. Liu. *Sentiment Analysis and Subjectivity.* in "Natural Language Processing", pages 627-666, Chapman & Hall, 2010.

[5] V. Cozza, A. Messina, D. Montesi, L. Arietta, M. Magnani. *Spatio-Temporal Keyword Queries in Social Networks.* In "Advances in Databases and Information Systems", pages 70-83, proceding of 17th East European Conference, ADBIS 2013, Genoa, Italy, September 2013.

[6] S.Russel and P. Norvig. *INTELLIGENZA ARTIFICIALE Un appoccio moderno.* cap. 1, pag 23 - 36, [edizione italiana a cura di Francesco Amigoni], Milano, Pearson Italia, 2010.

[7] A. Park, P. Paroubek. *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*, Proceding of LREC, 2010.

[8] C. D. Manning and H.Shütze. *Foundation of statistical natural language processing.* cap. 6,7,13,14, MIT press,1999

[9] A. Mukherjee, B. Liu, J. Wang, N. Glance, N. Jindal. *Detecting group review spam*. In 20th international conference companion on World wide web, pages 93-94, 2011.

[10] D. Maynard and A. Funk. *Automatic detection of political opinions in Tweet*. In "The Semantic Web", pages 88-99, Workshops ESWC, 2011.

[11] A. Go, R. Bhayani and L. Huang. *Twitter Sentiment Classification using Distant Supervision*. In CS224N Project Report, pages 1-6, Stanford, 2009.

[12] A. Mineo. *Una Guida all'utilizzo dell'Ambiente Statistico R*, Dipartimento di Scienze Statistiche e Matematiche, Università degli Studi di Palermo, 2003, `http://cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf` .

[13] F. Frascati. *Formulario di Statistica con R*, Università degli Studi di Firenze, 2008, `http://cran.r-project.org/doc/contrib/Frascati-FormularioStatisticaR.pdf` .

[14] I. Feinerer , K. Hornik. *Package "tm": Reference manual*, January 13 2014, `http://cran.r-project.org/web/packages/tm/tm.pdf` .

[15] P. Barbera. *Package "streamR": Reference manual*, January 7 2014, `http://cran.r-project.org/web/packages/streamR/streamR.pdf` .

[16] D. Kahle, H. Wickham. *Package "ggmap": Reference manual*, August 29 2013, `http://cran.r-project.org/web/packages/ggmap/ggmap.pdf` .

[17] L. Pantieri, T Gordini. *L'arte di scrivere in LaTeX*, Edizione 2012, `http://www.lorenzopantieri.net/LaTeX_files/ArteLaTeX.pdf` .

[18] F. Sebastiani. *Stanno Spiando le Nostre Emozioni? Sentiment Analysis nei Social Media*, Gruppo Human Language Technologies, Istituto di Scienza e Tecnologie dell'Informazione, CNR Pisa, `http://nmis.isti.cnr.it/sebastiani/SlideAreaAperta.pdf` .