Università di Pisa

Dipartimento di Filosofia, Letteratura e Linguistica

Corso di Laurea Magistrale in Informatica Umanistica

Tesi di laurea

Da Facebook a Twitter:
Creazione e utilizzo di una risorsa lessicale emotiva per la sentiment analysis di tweet

Candidato:
Alessandro Bondielli

Relatore:
Alessandro Lenci

Appello V

Anno Accademico 2015/2016

Sommario

La presente tesi di laurea si propone di esporre una ricerca effettuata in ambito di sentiment analysis ed emotion detection su testo prodotto da social network. In particolare ci si è proposti di costruire un sistema di classificazione automatica della polarità dei tweet, prendendo le mosse dal task di Sentiment Polarity Classification (SENTIPOLC) proposto da Evalita per la conferenza 2014. Il presente classificatore è stato sviluppato attraverso l'uso di un SVM in grado di tenere in considerazione sia feature lessicali e non lessicali presenti nel testo, sia la componente emotiva che il testo stesso presenta. Per la componente emotiva si è provveduto alla creazione di uno spazio distribuzionale emotivo partendo da dati testuali ricavati dalla piattaforma di social neworking Facebook.

A Cristina, ai miei genitori e ai miei amici,
perché senza di loro non sarei mai riuscito
a trovarmi dove mi trovo adesso.

Indice

Capitolo 1

Introduzione

Negli ultimi anni l'utilizzo di piattaforme di social media come Facebook e Twitter si è esteso a dismisura. Queste piattaforme sono utilizzate per diverse ragioni. Infatti, gli utenti spesso non si limitano a condividere contenuti tra amici e contatti utili, ma spesso considerano i social media come un utile mezzo di informazione.

Per molte persone i social network possono essere considerati la fonte primaria attraverso cui rimangono in contatto con il mondo e ottengono le ultime news. Un gran numero di studi recenti ha difatti fatto luce su come e quanto la crescente popolarità di blog, social media e altre attività della rete che prevedono contenuto generato da utenti abbia avuto impatto sulle modalità attraverso le quali le notizie vengono prodotte e riportate. Picard (2009) [57] afferma che i tool messi a disposizione dai social media garantiscono un modo semplice e poco costoso per i membri del pubblico di prendere parte a discussioni con gruppi di persone più ampi. Inoltre, ci sono forti prove a sostegno dell'esistenza di un legame tra il mondo dell'informazione e i social media per vasti gruppi di persone. È difatti stato stimato che nel 2013 circa il 30% della popolazione degli Stati Uniti ha ottenuto le notizie principalmente su Facebook (Holcomb et al., 2013)[13]. Una percentuale che chiaramente necessita di essere sommata a quella delle altre piattaforme, come Twitter e Youtube, e che è inoltre destinata a crescere nel corso degli anni insieme all'utenza attiva dei social network. Inoltre, la stessa ricerca afferma che la maggior parte degli utenti che ot-

tengono le notizie tramite piattaforme social tendono a fare affidamento su una sola di esse, mentre la percentuale di persone che si informano su due, tre o più siti cala drasticamente (60% per un solo sito, 26% per due e solo il 9% per tre o più).

Si può quindi affermare che l'informazione di massa si sta lentamente spostando verso piattaforme più generiche, mentre quelle ufficiali, come ad esempio siti specializzati in informazione, stanno perdendo la loro leadership. Come infatti affermato in Newman et al. (2012)[44], la crescente centralità di internet nella vita di tutti i giorni oltre che in ambito lavorativo ha fatto nascere molte domane riguardo le implicazioni per la produzione e il consumo di notizie. Come affermato nel report, nonostante una crescita nella popolazione totale in grado di fare uso attivamente di internet nell'arco degli anni 2009-2012 nel Regno Unito, questa non si riflette allo stesso modo nel consumo di testate giornalistiche online, mentre al contrario non mancano dati a sostegno della crescita nel consumo dei social media in tutto il mondo. Inoltre, la crescita nell'uso di social media e la loro rilevanza nella produzione e consumo di informazione e notizie, è evidente in un buon numero di casi specifici (Newman et al, 2012)[44].

Questo apparente declino delle piattaforme mediatiche più tradizionali, sia cartacee che sul web, secondo alcuni potrebbe essere collegato e condurre a un declino nella qualità e nella diversificazione della copertura mediatica (Chyi e Lasorsa, 2002)[17]. D'altra parte tuttavia, l'ascesa dei social network come piattaforme in cui news e informazione sono elementi essenziali promuove sicuramente un più fervido dibattito tra gli utenti, in quanto le notizie sono in grado di generare discussioni tra le reti sociali delle persone (Shah et al., 2005)[65]. Discussioni che sono disponibili pubblicamente per chiunque desideri unirsi o semplicemente osservare.

Questa tipologia di dato è proprio ciò che risulta interessante per la presente ricerca. Difatti, dietro ai commenti degli utenti su una notizia spesso si cela la loro opinione sull'argomento. Inoltre, il linguaggio utilizzato per esprimerle è generalmente colloquiale, e molto spesso ricco di termini ed espressioni di sdegno, offesa, rabbia,

e persino odio riguardo una determinata questione. Anche le emozioni positive sono in generale molto diffuse.

Recentemente anche l'interesse della comunità scientifica, e in particolare nel settore dell'elaborazione del linguaggio naturale, si è spostato sul web, grazie soprattutto alla possibilità di analizzare queste grandi quantità di contenuti, generati giornalmente dagli utenti e in buona parte pubblicamente accessibili.
La disciplina del *Sentiment Analysis*, ovvero l'analisi di sentimenti ed emozioni espressi attraverso il testo, sta infatti conoscendo un momento di grande popolarità, come testimoniano le numerose ricerche in merito presenti in letteratura. Risulta quindi importante offrire una panoramica riguardante gli scopi e i metodi di tale disciplina.

1.1 Sentiment Analisys

La *Sentiment Analisys*, o analisi dei sentimenti, è come già accennato l'area di ricerca che studia opinioni, sentimenti, valutazioni, attitudini ed emozioni delle persone riguardo entità e loro attributi espresse attraverso il testo scritto (Liu, 2012)[37]. Si tratta di un campo generalmente considerato un sottoproblema della tradizionale *elaborazione del linguaggio naturale* (NLP), sebbene alcuni autori tendano a individuarlo come un problema completo di elaborazione del linguaggio naturale su scala ridotta.
L'interesse per questo tipo di analisi in ambito di ricerca risulta essere relativamente recente, in quanto il termine è stato per la prima volta utilizzato in Nasukawa e Yi (2003)[42], sebbene i primi veri studi sull'argomento possano essere ricondotti, come osserva Liu (2013), alla fine degli anni '90. Il focus principale della materia è la ricerca della capacità di individuare, attraverso l'analisi del testo scritto, la tipologia di sentimento, generalmente considerato positivo, negativo o neutro, espresso dall'autore stesso del testo.

Le applicazioni pratiche di una disciplina come l'analisi dei sentimenti applicata direttamente al testo sono molteplici. Vista la crescita esponenziale di social media ed in generale modelli che pongono l'utente al centro della creazione dei contenuti presenti all'interno della rete, si riscontra un sempre maggiore interesse da parte di aziende e organizzazioni riguardo la possibilità di utilizzare questo tipo di dato a fini decisionali e strategici. "Che cosa pensano le altre persone" è da sempre stato una parte importante, se non fondamentale, per gran parte di ciascuno di noi nel processo decisionale, ben prima che la consapevolezza del World Wide Web fosse diffusa ovunque (Pang e Lee, 2008)[47], Con la successiva ascesa della popolarità di internet sempre più spesso ci si rivolge alla rete per avere informazioni riguardo un prodotto, le sue caratteristiche e le opinioni degli acquirenti, che possono non essere nè conoscenti personali nè professionisti del settore. In un panorama del genere molte aziende si trovano davanti alla necessità di analizzare la soddisfazione, o insoddisfazione della propria clientela, espressa sempre più spesso appunto, in forma scritta, attraverso la rete. Come sottolineato ancora da Pang e Lee (2008)[47], le applicazioni pratiche si estendono in vari campi, e un numero sempre crescente di aziende fa del sentiment analysis parte della propria missione di ricerca.
Alcune delle applicazioni pratiche della disciplina possono essere:

Siti di recensioni La possibilità di individuare i sentimenti espressi in una recensione rende più semplice effettuare motori di ricerca e di aggregazione di recensioni in base appunto ai sentimenti espressi in esse.

Sub componente tecnologica Le tecnologie di sentiment analysis possono dimostrarsi efficaci per migliorare soluzioni a problemi più complessi nell'ambito dell'NPL. Alcuni esempi possono essere il potenziamento di sistemi di raccomandazione, l'individuazione di *flame*, ovvero di discussioni estremamente accese che possono arrivare fino all'utilizzo di ingiurie e offese, soprattutto in ambito di social network, estrazione di informazione, in quanto scartare le informazioni presenti in testi soggettivi può migliorare le performances, question answering e summarization oltre a molti altri.

Businness e government intelligence In questo campo il sentiment analysis può risultare un ottimo alleato per gestire molti task di businness intelligence, come gestione della reputazione e public relations, oltre a predizioni del trend delle vendite di un prodotto.

Diversi domini Molti altri campi oltre all'informatica si sono interessati al sentiment analysis, con risultati spesso interessanti. Un esempio su tutti può essere chiaramente la politica, dove le opinioni della popolazione risultano essere fondamentali per gestire campagne elettorali.

Chiaramente queste sono solo alcune delle possibili applicazioni di una disciplina come il sentiment analysis, proprio perché le opinioni e i sentimenti sono parte integrante della vita di ciascuno, e spesso risultano essere il fattore fondamentale nei processi decisionali.

Nonostante la mole di studi sempre crescente effettuata in campo di sentiment analysis, è bene notare che i problemi posti dallo sviluppo di applicazioni simili sono tutt'altro che risolti. L'analisi infatti dei sentimenti all'interno del testo scritto pone difatti alcuni quesiti per i quali risulta difficile proporre risposte generali. Come ad esempio affermato in Pang e Lee (2008)[47], spesso i sentimenti e in generale l'espressione di soggettività è spesso molto sensibile al contesto, e ad un livello di granularità non molto fine, anche dipendente dal domino. Seppur vero infatti che la dipendenza da dominio risulta essere parzialmente causata da cambi nel vocabolario, anche la stessa identica espressione può indicare sentimenti differenti in domini diversi.

1.2 Emotion Detection

Un ulteriore aspetto, molto interessante ai fini della presente ricerca, riguardante un ambito più particolare del sentiment analysis, è sicuramente ciò che viene definito *Emotion Detection*. In questo caso, lo scopo è quello di riuscire ad individuare, all'interno del testo, le emozioni espresse dall'autore. Difatti, sebbene il riconoscimento delle emozioni e la loro analisi sia stato largamente studiato in vari campi, tra cui

neuroscienze, psicologia e scienze cognitive, interessa la possibilità di individuare le emozioni attraverso l'analisi di informazioni testuali (Binali e Potdar, 2012)[8]. Nonostante la presenza in letteratura di molte ricerche riguardo svariati problemi e potenziali applicazioni di sentiment analysis, non altrettanto si può affermare per quanto riguarda propriamente il task di emotion detection. Le ricerche in merito sono relativamente rare e soffrono della mancanza di prove empiriche (Shelke, 2014)[68], ovvero mancano di un solido background teoretico e applicativo, al contrario dei lavori eseguiti per quanto riguarda la sentiment analysis.

Tuttavia, risulta chiaro come il potenziale applicativo sia estremamente interessante. Come nel caso più generale della sentiment analysis difatti, sono numerose le possibili applicazioni soprattutto in ambito di strategie di marketing, ad esempio nel caso in cui un'azienda voglia rilevare le potenziali emozioni scaturite dall'acquisto di un prodotto, ma anche semplicemente dalla reazione a un video promozionale suscitata agli utenti dei vari social network. Il marketing emotivo infatti mira a stimolare emozioni, chiaramente positive, nei clienti in modo da legarli al brand e aumentare le vendite di prodotti e servizi. Oggigiorno non è il prodotto ad essere venduto, dato che per ogni categoria esiste una scelta pressoché smisurata. Il focus è sulla relazione che il cliente intraprende con il marchio e con le emozioni che il prodotto comunica (Shelke, 2014)[68]. Non solo tuttavia applicazioni di marketing e businness intelligence. L'uso di emotion detection, ancora una volta legato a doppio filo al testo generato attraverso i social, è stato proposto anche per questioni più strettamente sociali o di utilità, come ad esempio può essere considerata la rilevazione del cosiddetto *hate speech*. Nel capitolo 2 verranno presentate più in dettaglio assieme allo stato dell'arte alcune di queste applicazioni.

Nonostante l'importante applicabilità pratica di tale disciplina, anche in questo caso, ancor più che per il task più semplice di sentiment analysis, gli ostacoli per la realizzazione sono molti, primo fra tutti la difficile definizione di emozione, per la quale esistono differenti paradigmi e teorie da cui attingere. Anche in questo caso una descrizione dettagliata delle tecniche allo stato dell'arte per l'emotion detection verrà

presentata nel capitolo successivo. Tuttavia, è bene fin da subito porre l'accento sulle difficoltà pratiche imposte dal task, tra cui forse la più importante, e motivazione fondamentale per la presente ricerca, è la mancanza in molte lingue, tra cui l'italiano, di lessici emotivi accurati e sufficientemente ampi, così come di corpus taggati con i valori emotivi delle parole.

1.3 Motivazioni

Come già affermato il task di emotion detection si presta benissimo all'utilizzo su contenuti generati dagli utenti dei social network. Proprio in questo tipo di contenuto difatti si può trovare una tipologia testuale ricca di espressioni emotive e di opinioni sui più disparati argomenti.
La sfida posta dal testo in questione risulta quindi estremamente interessante. Numerosi studi infatti, soprattutto per quanto riguarda il task di classificazione della polarità di un post o tweet, sono presenti in letteratura, con risultati sicuramente incoraggianti. La questione del rilevamento di emozioni invece, per i problemi già presentati, e per la relativa novità del task, si potrebbe considerare ancora allo stato embrionale, sebbene siano stati presentati studi molto interessanti e con risultati altrettanto positivi (Cap 2). Il problema posto dall'identificazione dell'emozione è interessante come problema a sé stante, ma potrebbe, come è stato fatto in questa sede, essere affrontato come anche estensione del problema più generale posto dalla sentiment analysis.

Nello specifico, lo scopo della presente ricerca è infatti quello di valutare i possibili miglioramenti nelle prestazioni di un classificatore di polarità di tweet se ad esso vengono affiancate una serie di feature emotive ricavate direttamente dal testo.
Si è deciso quindi di partire dal classificatore svilupatto dal Laboratorio di Linguistica Computazionale dell'Università Di Pisa (CoLingLab) nell'ambito del task di sentiment analysis proposto per la conferenza Evalita 2014 (Passaro et al., 2014)[53]. Il classificatore, come sarà spiegato nei capitoli successivi, prevede l'utilizzo di un

SVM addestrato su una serie di feature lessicali e non lessicali estratte dal corpus di addestramento fornito da Evalita. È stato quindi deciso di integrare le feature del classificatore originale con una serie di feature emotive.

Per individuare il testo emotivo è stato deciso che sarebbe stato interessante creare una risorsa emotiva ad hoc. In questo caso, si è convenuto di creare la risorsa partendo proprio da dati estremamente emotivi, come sono quelli già menzionati ricavabili dall'interazione degli utenti attraverso i social network. È stato quindi scaricato un corpus di post e commenti di Facebook, i cui token sono stati utilizzati per creare uno spazio distribuzionale emotivo, assegnando ad ogni parola target un valore per ciascuna delle emozioni scelte.

Una volta creata la risorsa il classificatore è stato quindi aggiornato con una serie di feature emotive direttamente estraibili dal corpus, e ne sono state valutate le performance.

La presente tesi è organizzata come segue. Nel capitolo successivo sarà analizzato lo stato dell'arte delle discipline sopra citate, dalla tradizionale sentiment analysis all'emotion detection, considerando brevemente anche la creazione di spazi distribuzionali specificamente creati per scopi di individuazione emotiva, oltre ad una panoramica sulla classificazione automatica attraverso *Support Vector Machines (SVM)*. In seguito sarà presentato l'effettivo processo di acquisizione del corpus utilizzato per creare lo spazio distribuzionale. Infine verranno mostrati i risultati ottenuti tramite gli esperimenti effettuati sul classificatore, dopo aver presentato il sistema di estrazione delle feature scelte.

Capitolo 2

Stato dell'arte

Nel presente capitolo verrà analizzato lo stato dell'arte attuale per le discipline interessate da questa ricerca. In particolare, ci si soffermerà sulle tecniche specialmente di sentiment analysis, in quanto focus principale degli esperimenti svolti. Si cercherà inoltre di offrire una panoramica sufficientemente dettagliata riguardante anche il campo dell'emotion detection, utile ad evidenziare come sempre più spesso si senta il bisogno di risorse emotive da utilizzare nelle più svariate applicazioni, sia di ricerca che commerciali. Verranno successivamente descritte in breve le potenzialità offerte dalla creazione di spazi distribuzionali, in particolare per quanto riguarda l'emotività delle parole. Infine, saranno esplorate le tecniche di classificazione attraverso Support Vector Machines, utilizzate nella fase finale della presente ricerca.

2.1 Sentiment Analysis

Come già evidenziato nel capitolo introduttivo, il problema posto dall'analisi dei sentimenti presenti in un testo e della polarità espressa è sicuramente al giorno d'oggi uno dei più discussi e interessanti nel panorama della linguistica computazionale e della elaborazione del linguaggio naturale. Sebbene l'interesse sia infatti piuttosto recente, rispetto soprattutto ai primi studi in campo di elaborazione del linguaggio naturale, è chiaro come questo tipo di analisi possa risultare interessante sotto vari punti di vista. Innanzitutto, come già affermato, risulta essere adattabile a un

numero pressoché illimitato di applicazioni, in ogni dominio. Soprattutto a livello commerciale sono sempre di più le aziende interessate a sviluppare sistemi in grado di esprimere automaticamente la soddisfazione del cliente. Inoltre, questo tipo di approccio prevede la soluzione di sfide computazionali interessanti, molte delle quali ancora non erano state affrontate. Infine, per la prima volta nella storia dell'umanità si riscontra la disponibilità di un'enorme quantità di dati contenenti opinioni sui social media del web (Liu, 2012)[37].

Come spiegato in Liu (2012)[37], generalizzando si possono distinguere tre livelli di analisi per quanto riguarda i sentimenti e le opinioni individuabili nei testi:

Livello dei documenti. In questo caso, il focus è quello di classificare la polarità di un intero documento, e distinguere se esprime un'opinione positiva o negativa (Pang et al., 2002[48]; Turney, 2002[74]). Un esempio molto semplice potrebbe essere la recensione di un prodotto, che può essere considerata positiva o negativa. Chiaramente tuttavia, questo livello di analisi presuppone che il testo in questione, indipendentemente dalla lunghezza, esprima solamente un'opinione. Questo livello quindi non risulta adatto a compiti che prevedono l'individuazione di opinioni per più prodotti, o più in generale per testi che non esprimono una singola opinione.

Livello delle frasi. In questo caso l'analisi si presenta con una granularità più elevata, in quanto il focus non è più sul documento nel suo insieme ma sulle singole frasi. Liu (2012)[37] suggerisce ancora come questo tipo di analisi sia strettamente correlato al task di classificazione di soggettività, presentato ad esempio in Wiebe e O'Hara (1999)[83]. Tuttavia viene fatto anche in questo caso notare come non sia certo che la soggettività di una frase equivalga all'espressione di un qualche tipo di opinione o sentimento.

Livello dell'aspetto e dell'entità. Il livello di granularità più elevato infine si propone di analizzare direttamente l'opinione stessa, anziché le costruzioni linguistiche, individuando quindi per ogni opinione un proprio target a cui fa

riferimento. Un'analisi del genere chiaramente è in grado di distinguere un maggior numero di opinioni e rispettivi target all'interno del testo, producendo ad esempio un riassunto strutturato di opinioni riguardo entità e loro aspetti (Liu, 2012)[37].

Nel presente caso, si è scelto di focalizzare l'attenzione sul livello di classificazione dei documenti, in quanto il task di partenza che ha ispirato questa ricerca richiede proprio questo tipo di analisi.

La classificazione della polarità di un documento, spesso riferita come *document level sentiment classification*, in quanto il livello con granularità meno fine, risulta essere un compito relativamente più semplice ma allo stesso tempo molto interessante e fruttuoso, e in quanto tale ha attratto un grande numero di ricerche.
In generale, Liu (2012)[37] definisce il problema come segue.
Innanzitutto, un'*opinione* in generale è considerata come una quintupla
$(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$
dove e_i è il nome di un'entità, a_{ij} è un aspetto di e_i, s_{ijkl} è il sentimento riguardante a_{ij} dell'entità e_i, h_k è chi detiene l'opinione, e t_l è il tempo in cui l'opinione è espressa da h_k. Il sentimento s_{ijkl} può essere positivo, negativo o neutro, o espresso con diversi livelli di forza e intensità.
A questo punto si può definire il problema dell'identificazione, ad esempio nel contesto di una recensione, in questo modo:

> Dato un documento d volto a valutare un'entità, si determini il sentimento generale s dell'autore riguardo l'entità, cioè si determini s espresso sull'aspetto generale nella quintupla
> $(, GENERAL, s, ,)$
> dove l'entità e, l'autore h, e il tempo t in cui è stata espressa l'opinione sono considerati conosciuti o irrilevanti. Esistono due formulazione in base ai valori assunti da s. Se assume valori categorici, ad esempio positivo o negativo, si tratta di un problema di classificazione. Se s assume

valori numerici o ordinali dato un intervallo, si tratta di un problema di regressione.

Inoltre è importante notare come, per rendere questo tipo di task fattibile in pratica, è necessario assumere a priori, come sottolineato in Liu (2010)[36], che il documento d esprima opinioni su una singola entità e e che questa opinione sia espressa da una singola persona h.

Il task di classificazione di documenti in base all'opinione espressa può essere eseguito attraverso due approcci principali: *apprendimento supervisionato* e *apprendimento non supervisionato*.
Per quanto riguarda i metodi di apprendimento supervisionato, senza dubbio si tratta dell'approccio maggiormente utilizzato in letteratura, ed è stato anche quello utilizzato nella presente ricerca e dalla ricerca a cui questa tesi fa capo.
Solitamente, come già accennato ed espresso in Liu (2012)[37], il problema di classificazione di documenti in base alla loro polarità si presenta come un problema di classificazione a 2 classi, ovvero positivo e negativo. La maggior parte degli approcci infatti non fa uso di una classe neutra, che renderebbe in generale il task molto più complesso. Per fare un esempio, se si considerassero le recensioni online come dati di addestramento, tipicamente viene assegnato un valore di negatività alle recensioni con 1 o 2 stelle, e al contrario un valore di positività alle recensioni da 4 o 5 stelle. Il valore di neutralità in questo caso potrebbe essere assegnato alle recensioni che hanno assegnato al prodotto 3 stelle.
Come inoltre enfatizzato in Liu (2012)[37], il problema così posto di sentiment classification risulta essere essenzialmente un problema di classificazione testuale. Tuttavia in questo caso, parole che indicano sentimenti ed opinioni sono considerate più importanti rispetto alle parole relative all'argomento oggetto del documento.
Proprio per la sua natura testuale, al task di sentiment classification può essere adattato e applicato un qualsiasi modello di apprendimento supervisionato, come ad esempio Naive Bayes Classification, Decision Tree, o Support Vector Machines (SVM) (Joachims, 1999[29]; Shawe-Taylor e Cristianini, 2000[67]), come nel caso

della presente ricerca. La prima ricerca a intraprendere questo approccio per la classificazione, in particolare per quanto riguarda recensioni di film suddivise in positive e negative, è stata condotta da Pang et al. (2002)[48].

Una delle caratteristiche principali degli approcci supervisionati, soprattutto per quanto riguarda la classificazione testuale, ma in generale qualsiasi tipo di classificazione, è la scelte delle feature. Difatti, come sottolineato in Pang e Lee (2008)[47], convertire un testo in un vettore di feature o in un qualunque altro tipo di rappresentazione che renda le caratteristiche più importanti e salienti del testo disponibili è una parte fondamentale degli approcci guidati dai dati per la classificazione del testo. Sebbene sia presente una mole smisurata di lavori riguardanti la selezione di feature per gli approcci di machine learning in generale, per rimanere nell'ambito della sentiment analyisis si possono ad esempio elencare:

- Presenza di termini o frequenza
- Part-Of-Speech
- Parole che esprimono sentimenti
- Dipendenze sintattiche
- Valence Shifters (ad esempio negazioni, intensificativi e diminutivi)
- Feature riguardanti termini oltre gli unigrammi
- Feature riguardanti l'argomento

Pang e Lee (2008)[47] fanno inoltre notare quanto sia di fondamentale importanza la possibilità di ottenere testo annotato. Difatti, la proliferazione di risorse contenenti documenti soggettivi ha reso possibile la valutazione anche empirica della bontà dei sistemi prodotti, come non era stato possibile durante i primi anni di interesse verso la materia. Il testo annotato è stato fondamentale nel rendere possibili approcci statistici, come la maggior parte di quelli diffusi al giorno d'oggi. In particolare, l'aumento sia nella quantità che nella qualità di dati annotati per quanto riguarda specificamente l'analisi dei sentimenti nel testo, ove l'annotazione può essere sia

frutto di lavoro manuale iniziato da parte dei ricercatori sia generata attraverso altri metodi, è stato un fattore essenziale per il proliferare di ricerca sia per quanto riguarda l'apprendimento supervisionato che non supervisionato.
Grazie allo sviluppo di questo tipo di risorse, combinato con un grande interesse da parte della comunità scientifica, è presente un gran numero di ricerche in letteratura, come evidenziato in Liu (2012)[37], a cui si rimanda per una trattazione e una rassegna più completa delle ricerche.

Per quanto riguarda invece gli approcci basati su metodi non supervisionati, lo scopo di classificazione rimane identico, ma le modalità per ottenerlo divergono. In questo caso, molto spesso si cerca in primo luogo di creare in maniera non supervisionata dei lessici contenenti un gran numero di parole ed espressioni considerate polarizzate, per poi determinare il grado di polarità di un'unità testuale utilizzando funzioni basate sugli indicatori positivi e negativi (o più semplicemente soggettivi) individuati all'interno di esso, come determinato dal lessico (Pang e Lee, 2008)[47].
Un altro possibile approccio è definito *bootstrapping*, in quanto prevede di utilizzare il risultato di un classificatore a disposizione per creare dei dati annotati, sui quali successivamente applicare un algoritmo di apprendimento automatico.
Infine, un'altro esempio può essere fornito da Turney (2002)[74], nel quale è mostrato come sia possibile utilizzare per la classificazione dei pattern sintattici che hanno alta probabilità di essere usati per esprimere opinioni. Anche in questo caso si rimanda a Liu (2012)[37] e Pang e Lee (2008)[47] per una rassegna più approfondita delle varie tecniche proposte nel campo delle tecniche di apprendimento non supervisionato.

Oltre alle ricerche considerate, come già mostrato è chiaro comunque come le modalità di applicazione di una disciplina come il sentiment analysis non si limitino all'individuazione della polarità dei documenti. Difatti, è evidente la necessità di valutare un diverso tipo di dato soggettivo, come ad esempio testi di blog e news, che differisce dalla recensione considerata finora come esempio. Va considerata infatti l'esistenza di altri task di sentiment analysis, basati ad esempio sull'aspetto del singolo termine o sulla polarità delle singole frasi all'interno di un documento, che

prevedono approcci anche diversi tra loro. Tuttavia, si è deciso di soffermarsi proprio su questo aspetto dell'analisi in primo luogo per il carattere del task proposto come base di partenza per la presente ricerca, volto appunto a questo tipo di classificazione, e in secondo luogo in quanto si ritiene che, coerentemente con l'analisi proposta, una tipologia di documento come il tweet, considerato il numero estremamente ridotto di caratteri su cui può fare affidamento, sebbene in grado ovviamente di esprimere più opinioni, può risultare più interessante e facilmente accessibile in un ottica globale.

2.2 Emotion Detection

Nel capitolo di apertura è stata introdotta la disciplina che prende il nome di *emotion detection*, che a differenza del task di sentiment analysis e classification, può esserne considerato per certi versi un sottoproblema e ha in particolare la finalità di individuare le emozioni presenti nel testo. Come è stato già chiarito, questo tipo di approccio, sebbene ancora la letteratura in merito non sia particolarmente ampia, oltre a risultare molto fruttuoso in campo applicativo, soprattutto in settori come quello del marketing e finanziario, ma anche educativo, di sicurezza pubblica e molti altri, propone delle sfide estremamente interessanti per i ricercatori. Sono difatti molti i punti interrogativi e le diverse proposte dei ricercatori in merito all'approccio più corretto.
Per la successiva trattazione dell'argomento si farà in gran parte riferimento a Binali e Potdar (2012)[8], a cui si rimanda per informazioni più dettagliate.

In primo luogo, risulta molto difficile dare una definizione completamente esaustiva e oggettiva di cosa sia un'emozione, o cosa siano le emozioni in generale. Esistono infatti in letteratura, tre approcci principali per modellare le emozioni: *categorico*, *dimensionale* e *appraisal-based*, ovvero basato su valutazioni(Gunes e Pantic, 2010)[23]. In ognuno dei casi, i ricercatori, spesso in ambito di scienze psicologiche o neurolinguistiche, sono guidati dall'assunzione che le persone abbiano dei meccanismi interni per un insieme limitato di reazioni (tipicamente felicità, rabbia, tristezza,

paura, disgusto e interesse) i quali, una volta azionati, possono essere misurati in maniera chiara e oggettiva (Barret, 2006)[4].

L'approccio categorico, sicuramente il più immediato ma comunque molto interessante, si basa appunto sulla modellazione dello spettro emotivo in classi di emozioni distinte, rappresentate ad esempio da una serie di emozioni di base, ritenute come esprimibili in qualsiasi lingua. Due esempi possono essere le 6 emozioni base di Ekman (1992)[21], ovvero felicità, tristezza, paura, sorpresa, rabbia e disgusto, o l'inventario emotivo di 8 emozioni teorizzato da Plutchik (1994)[59], composto da: GIOIA, TRISTEZZA, RABBIA, PAURA, FIDUCIA, DISGUSTO, SOPRESA E ATTESE. Si fa notare come l'inventario emotivo di Plutchik sia quello considerato anche dalla presente ricerca, in particolare per quanto concerne la creazione dello spazio emotivo.
Per quanto riguarda la parte di effettiva classificazione del testo, l'approccio categorico quindi cerca di assegnare il testo a una specifica categoria emotiva, in base al modello adottato. Il risultato può essere ottenuto usando approcci di varia natura, sia regole modellate manualmente che approcci basati sul machine learning (Binali e Potdar, 2010)[9].

Per quanto concerne invece l'approccio dimensionale, vengono considerati i vari stati emotivi non come opposti ed indipendenti l'un l'altro, ma al contrario come in relazione l'uno con l'altro (Binali e Potdar, 2012)[8]. Le emozioni sono in questo caso descritte come aventi un piccolo set di dimensioni in grado di descriverle, ovvero *valence*, *arousal* e *dominance* (Jin e Wang, 2005)[28]. In questo modo è più semplice tracciare delle relazioni tra le varie emozioni in base alle dimensioni sopra citate, e soprattutto si rende possibile individuare meglio testi che esprimono più di una emozione, o emozioni miste.

L'approccio appraisal-based infine, si può considerare come una evoluzione dell'approccio categorico. Vengono infatti utilizzate le nozioni della teoria dell'appraisal, con modelli emotivi componenziali. I modelli basati su appraisal rappresentano le di-

mensioni del significato che sono associate con emozioni particolari (Barret, 2006)[4]. Questo tipo di approccio considera le emozioni come espresse, e quindi individuabili, attraverso le modifiche in tutte le componenti rilevanti, tra cui sono da includere cognizione, motivazione, reazioni fisiologiche, motorie e sentimenti espressi (Gunes e Pantic, 2010)[23].

Chiaramente ognuno degli approcci sopra citati possiede delle caratteristiche peculiari che lo rendono più o meno efficace a seconda del task richiesto. Come già accennato infatti, mentre l'approccio categorico risulta più semplice, ha difficoltà nel rilevare emozioni miste, come invece si è più facilmente in grado di fare seguendo un approccio di tipo dimensionale. L'approccio appraisal based infine, sebbene utilizzi una vasta gamma di elementi per definire la risposta emotiva, tende in ultima analisi a distinguere le emozioni basandosi su categorie.

Una volta scelto l'approccio da seguire, è importante inoltre identificare e distinguere i vari modelli proposti per effettuare l'individuazione delle emozioni nel testo. Si possono generalmente distinguere tre tipologie di modelli principali.

I *modelli basati sull'utilizzo di un corpus* fanno uso di un lessico emotivo pesato ricavato da documenti di training annotati, attraverso il quale in seguito viene costruito il modello di predizione delle emozioni (Binali e Potdar, 2012)[8]. Spesso per la creazione del lessico emotivo, come si vedrà nella sezione riguardante lo spazio distribuzionale, si utilizzano algoritmi diversi ma molto spesso basati sul principio secondo cui il significato di una parola, o in questo caso il suo contenuto emotivo, può essere indotto dall'osservazione statistica del suo utilizzo in un campione significativamente grande di una lingua (Shelke, 2014)[68].
In particolare, viene fatto notare come la letteratura riguardante il task di disambiguazione semantica delle parole possa venire in aiuto a questo tipo di modelli per quanto concerne gli indicatori delle proprietà delle parole, ad esempio le parole circostanti e la loro Part-Of-Speech, collocazioni, parole chiave, bigrammi, entità nominate, feature sintattiche e relazioni semantiche con le altre parole nel contesto

(Keshtkar e Inkpen, 2010)[31]. Grazie a ciò ad esempio Keshtar e Inkpen (2010)[31] hanno creato 18 feature volte a individuare i pattern emotivi, e attraverso un algoritmo di bootstrapping sono stati in grado di creare parafrasi relative alle emozioni rappresente e pattern di estrazione delle stesse, con risultati soddisfacenti.
Un ulteriore esempio è dato da Strapparava e Mihalcea (2008)[71], che attraverso l'utilizzo di analisi sintattiche profonde sono riusciti a ottenere ottimi risultati, soprattutto andando a considerare un alto livello di granularità per le emozioni individuate.

I modelli che prevedono l'uso di *tecniche di machine learning* invece fanno generalmente uso, come caratteristica fondamentale di un corpus annotato con i valori emotivi, necessario per addestrare un classificatore emotivo. L'addestramento in questo caso può essere sia supervisionato che non supervisionato (Binali e Potdar, 2012)[8]. Proprio la necessità di una risorsa già annotata può in questo caso essere un ostacolo. Difatti, essendo l'emotion detection un campo di studio relativamente nuovo, questo tipo di risorse scarseggiano, e anzi se ne avverte un importante bisogno. Burget et al. (2011)[11] hanno proposto un metodo che consiste in larga parte di una fase di pre-processing del testo e successivamente di etichettatura tramite un classificatore, utilizzando la misura TF-IDF (Term Frequency-Inverse Document Frequency) per calcolare la rilevanza delle parole per ciascuna emozione. Un altro interessante approccio, come mostrato in Albornoz et al. (2010)[1], prevede l'utilizzo dell'approccio categorico alle emozioni, utilizzando tecniche di pre processing e una risorsa lessicale come WordNet Affect per definire l'appartenenza dei termini ad una specifica emozione. Come in gran parte dei task di classificazione che fanno utilizzo di tecniche di machine learning, le misure di valutazione sono *Precision*, *Recall* e *F-measure*. In paticolare, ottimi risultati sono stati ottenuti da Yassine e Hajj (2010)[86], che utilizzando la tecnica di 10-fold cross validation in un classificatore SVM, in aggiunta a lessici creati ad hoc per distinguere acronimi ed emoticon nella parte social del web, ha raggiunto un'accuratezza dell'87% sull'individuazione della soggettività di documenti.

Un ulteriore distinzione va effettuata, tra esperimenti riguardanti la classificazione delle emozioni a livello di parole e a livello di frasi. Entrambe sono state studiate da Das e Bandyopadhyay (2009)[18]. L'annotazione a livello di parole è stata effettuata utlizzando WordNet Affect (Valitutti, 2004)[77] e SentiwordNet (Baccianella et al., 2010)[3]. La prima è una risorsa estensione di WordNet, in cui sono presenti synset in grado di rappresentare concetti affettivi(emotivi) correlati con parole emotive, mentre la seconda è una risorsa propriamente lessicale che assegna a ogni synset di Wordnet tre valori di positività, negatività e oggettività. In seguito ogni frase è stata sottoposta a un sistema di valutazione basato sui sensi delle parole che la compongono.
Viene infine fatto notare da Binali e Potdar (2012)[8] come buona parte degli studi considerati facciano uso di un classificatore di tipo SVM o CRF (Conditional Random Field), mentre alcuni esplorano le possibilità offerte dall'utilizzo di entrambi o da più classificatori, senza ottenere tuttavia prestazioni significativamente differenti, come nel caso di Quan e Ren (2010)[60].

Infine, i modelli sviluppati attraverso l'uso di *Knowledge-based techiniques* sono soliti applicare regole linguistiche sfruttando la conoscenza delle strutture sintattiche in congiunzione con l'utilizzo di risorse lessicali specificamente pensate per descrivere sentimenti ed emotività, come ad esempio Word-Net Affect o SentiWordNet (Binali e Potdar, 2012)[8].
Molto spesso viene utilizzato un lessico per la creazione di feature atte a una classificazione emotiva, come in Yang et al. (2007)[85]. Difatti, in letteratura, sono svariati i lessici proposti per questo tipo di task. Un esempio può essere in questo caso il lavoro svolto da Neviarouskaya et al. (2010)[43], nel quale è stato proposto un lessico di acronimi sviluppato appositamente per andare a risolvere i problemi generati dall'utilizzo di linguaggio di tipo informale soprattutto nei social media. Un ulteriore esempio molto calzante, soprattutto per la presente ricerca, è la risorsa ITEM, per l'italiano, sviluppata dal ColingLab (Passaro et al., 2015)[54]. In questo caso è stata seguita la procedura presentata in Mohammad e Turney (2013)[41] per crea-

re attraverso il crowdsourcing una selezione di lemmi estremamente associati a una delle 8 emozioni proposte da Plutchik, ed in seguito estenderla attraverso l'utilizzo di un corpus e tecniche di semantica distribuzionale. L'approccio verrà descritto più dettagliatamente in seguito, in quanto parte integrante della presente ricerca.

In conclusione, è chiaro come, a differenza di altri task proposti nell'ambito dell'elaborazione del linguaggio naturale, l'individuazione delle emozioni all'interno del testo sia sicuramente ancora dibattuto, sia per quanto riguarda i metodi che per quanto riguarda i risultati ottenuti.

2.3 Vector Space Models

In questa sezione verranno in breve esplorate le possibilità e le caratteristiche dei *modelli a spazi vettoriali* (Vector Space Models, VSM). Pur non rappresentando il focus principale della presente ricerca, sono stati utilizzati per la creazione della risorsa emotiva utilizzata, e risulta necessario fornire una descrizione dell'argomento. Per una rassegna più completa e accurata si rimanda a Turney e Pantel (2010)[51].

Un punto fermo fondamentale per la letteratura riguardo l'elaborazione del linguaggio naturale è sicuramente l'idea che i computer abbiano una gran difficoltà nella comprensione del linguaggio umano. Proprio per questo motivo è fondamentale lo sviluppo di tecnologie che riescano a colmare lo spazio tra il significato delle parole e una effettiva rappresentazione computazionale dello stesso. Parte di queste tecnologie è appunto rappresentata dai Vector Space Models, per i quali sono evidenti promettenti segni di sviluppo, anche grazie alla loro relazione con le ipotesi distribuzionali.

L'ipotesi distribuzionale difatti, proposta per la prima volta da Harris (1968)[26], afferma che parole simili tendano ad apparire in simili contesti. L'applicazione computazionale di questo paradigma prevede quindi l'utilizzo di algoritmi che misurino la similarità di significato utilizzando vettori, matrici, e tensori di ordine più elevato

(Turney e Pantel, 2010)[51]. Ogni parola è quindi associata a un vettore di feature pesate, le quali generalmente corrispondono ad altre parole che co-occorrono con la parola caratterizzata nel contesto. La similarità distribuzionale è quindi misurata quantificando il grado di similarità tra coppie di vettori di questo tipo (Zhitomirsky-Geffet e Dagan, 2009)[88]. La novità fondamentale introdotta dall'uso dei modelli semantici vettoriali è data quindi dalla capacità di estrarre informazione semantica utilizzando le informazioni fornite dalla frequenza dei termini in un corpus testuale (Turney e Pantel, 2010)[51].

Un altro importante fattore da tenere in considerazione, soprattutto per lo sviluppo pratico della presente ricerca, è ciò che nell'ambito delle scienze cognitive viene chiamato *teoria dei prototipi*, per la quale viene molto spesso fatto uso di vettori di feature. In questo caso, la teoria, come proposto in Rosch e Lloyd (1978)[63] e Lakoff (1987)[32], si sviluppa dall'idea di base che alcuni oggetti siano più centrali per una categoria rispetto ad altri. Per fare un esempio classico, un pettirosso è più centrale di un pinguino per la categoria *uccello*. Allo stesso modo, anche la parola "terrore" è sicuramente più centrale della parola "asciugamano" per quanto riguarda l'emozione *paura*.

La trattazione di Turney e Pantel (2010)[51] suddivide i tipi di spazi vettoriali per la semantica in base al tipo di matrice utilizzata: *matrice Termine-Documento*, *matrice Termine-Contesto*, *matrice Coppia-Pattern*. In questa sede ci si soffermerà in particolare sull'utilizzo di matrici Termine-Contesto, in quanto elemento centrale per la creazione dello spazio distribuzionale utilizzato.
In ognuno dei casi, il substrato teorico a supporto dell'utilizzo di vettori e matrici è garantito dall'*ipotesi statistica della semantica*: pattern statistici riguardanti l'utilizzo delle parole possono essere utilizzati per individuare il significato delle espressioni (Turney e Pantel, 2010)[51].

2.3.1 Tipologie di matrici

Matrice Termine-Documento Nel caso di matrici termine-documento, lo scopo risulta quello di individuare la similarità tra documenti in relazione agli elementi, ovvero le parole, che ne fanno parte. In generale, le righe della matrice corrispondono alle parole, mentre le colonne ai documenti a disposizione. All'interno della matrice, ogni cella rappresenta la presenza o meno di un determinato termine all'interno di un determinato documento. Chiaramente, la matrice risultante sarà sparsa, in quanto per ogni termine e documento deve essere rappresentata la presenza o assenza di una co-occorrenza fra i due elementi.

In una matrice di questo tipo, i vettori rappresentano i corrispondenti documenti attraverso l'ipotesi della *bag of words* (Turney e Pantel, 2010)[51]. L'ipotesi, come stipulato da Salton et al. (1975)[64], afferma che le frequenze delle parole presenti all'interno di un documento sono in grado di indicare la rilevanza del documento stesso per una determinata query. Se documento e query hanno vettori colonna simili (cioè sono presenti parole simili con frequenze simili) in una matrice termine-documento, tendono ad avere significati e contenuti simili.

Per ogni documento quindi non è rappresentata l'informazione riguardante l'ordine delle parole nel documento. Viene invece registrata la presenza o assenza di ogni termine e la sua frequenza all'interno del documento. Ad esempio, se la parola "argento" è contenuta nel documento D_1 mentre la parola "classe" è contenuta nel documento D_2, la matrice riporterà per ogni riga le parole presenti nei documenti, per ogni colonna i documenti considerati e all'interno della matrice i valori che rappresentano la presenza o assenza di una determinata parola in un determinato documento. Sebbene dunque l'ordine sequenziale delle parole vada perduto, e con esso la struttura sintattica pertinente, strutture di questo tipo, implementate in motori di ricerca, funzionano sorprendentemente bene. Una giustificazione intuitiva per questo comportamento può essere che l'argomento di un documento influenzi probabilisticamente la scelta delle parole che l'autore utilizza per comporlo (Turney e Pantel, 2010)[51]. Questa tipologia di matrici viene quindi utilizzata spesso in task di *information retrieval*, come document retrieval, classification, clustering e

segmentation.

Matrice Termine-Contesto Le matrici di tipo termine-contesto sono nate dalla volontà di essere in grado di misurare la similarità semantica tra parole, anziché tra documenti. L'intuizione è realizzata in Deerwester et al. (1990)[19], in cui è osservato come lo spostamento di focus necessario all'individuazione della similarità tra termini può essere effettuato semplicemente andando a calcolare la similarità tra i vettori riga di una matrice termine-documento anziché tra le colonne. Come precisato in Turney e Pantel (2010)[51], l'ispirazione per il lavoro di Deerwester et al. (1990)[19] è chiaramente il lavoro svolto da Salton et al. (1975)[64]. Tuttavia in questo caso si è notato come i documenti non fossero l'unità di misura più adeguata a calcolare la similarità tra i singoli termini. Richiamando ancora una volta l'ipotesi distribuzionale formulata da Harris (1954)[25], parole simili dal punto di vista semantico tendano a comparire in contesti simili. Viene quindi trattato il contesto come uno pseudo-documento da utilizzare nella matrice.
In questo caso dunque, per ogni riga sono riportati i termini target, mentre in ogni colonna sono rappresentati i contesti da cui questi termini sono stati estratti, ad esempio una finestra che va dalle due parole precedenti alle due parole successive (termine target escluso). I valori rappresentano ancora una volta la presenza o meno all'interno del corpus di occorrenze della parola target all'interno di ognuno dei contesti. Nel corso degli anni sono state proposte e adottate varie soluzioni per quanto concerne la rappresentazione del contesto. Turney e Pantel (2010)[51] mostrano come ci si possa riferire ad esso come a una finestra di parole che compaiono prima e dopo il termine target, come ad esempio in Lund e Burgess (1996)[39], ma anche nel lavoro svolto per la presente ricerca. Ancora il contesto può essere considerato come formato da dipendenze grammaticali (Lin, 1998[34]; Padò e Lapata, 2007[46]).
L'utilizzo pratico anche in questo caso risulta molto interessante. Le matrici termine-contesto possono infatti essere utilizzate per vari scopi. Anzitutto per misurare la similarità tra le parole come teorizzato da Deerwester et al. (1990)[19]. Ma anche clustering di parole (Pereira et al., 1993)[56], classificazione (Turney e Littman,

2003)[75], generazione automatica di thesauri (Fellbaum, 1998)[20], word-sense disambiguation (Agirre e Edmonds, 2006[**?**]; Pedersen, 2006[55]), estrazione di informazione (Pasca et al., 2006)[52] e molto altro. Nel capitolo 4.1 sarà mostrato un utilizzo pratico, in particolare per la creazione della risorsa emotiva utilizzata per la classificazione a partire da un corpus di post e commenti ricavati da Facebook.

Matrice Coppia-Pattern Le matrici coppia-pattern rappresentano invece un'informazione di tipo differente. Sono state introdotte per la prima volta in Lin e Pantel (2001)[35], dove è stata proposta *l'ipotesi distribuzionale estesa*, secondo cui pattern che co-occorrono con coppie di parole simili tendono ad avere significati simili. Nella pratica l'intuizione è sviluppata tramite la creazione di una matrice in cui i vettori riga corrispondono a coppie di parole, mentre i vettori colonna rappresentano i pattern in cui queste coppie di parole co-occorrono (Turney e Pantel, 2010)[51]. Per fare un esempio, le coppie di parole *sega:legna* e *martello:chiodo* possono corrispondere ai pattern *X taglia Y* e *X batte Y*.
Nella matrice è quindi rappresentata una coppia per ogni riga, e un pattern per ogni colonna. I valori all'interno della matrice rappresentano la presenza all'interno del corpus dello specifico pattern con la specifica coppia di termini. La similarità è quindi misurata sui vettori colonna. Tuttavia, Turney et al. (2003)[76], attraverso la misurazione della similarità dei vettori riga, hanno cercato di misurare la similarità semantica delle relazioni tra coppie di parole (Turney e Pantel, 2010)[51].
Anche per questo tipo di spazi vettoriali le applicazioni sono molteplici, in particolare per quanto riguarda le relazioni. In letteratura sono presenti, oltre all'individuazione di similarità tra coppie di parole e pattern, anche esempi di clustering (Bicici e Yuret, 2006)[7], classificazione (Chklovski e Pantel, 2004)[15] e ricerca di relazioni (Cafarella et al., 2006)[12], oltre anche alla generazione automatica di thesauri (Snow et al., 2006)[70].

Tensori di ordine superiore Oltre alle matrici, nel corso degli anni sono state sviluppate modelli facenti utilizzo di tensori di ordine superiore a quello delle matrici, allo scopo di individuare differenti tipi di relazione. Un esempio può essere trovato

in Chew et al. (2007)[14], in cui è stato sviluppato un tensore di terzo ordine di tipo termine-documento-lingua allo scopo di effettuare task di information retrieval su più lingue (Turney e Pantel, 2010)[51].

2.3.2 Costruzione dello spazio

Prima della costruzione di una matrice a partire da un corpus testuale, risulta spesso necessario effettuare una fase di preprocessing linguistico. Utile infatti è la tokenizzazione del testo, come anche uan fase di annotazione per Part-Of-Speech, ad esempio per filtrare parti del discorso non necessarie all'analisi, parsing e word-sense tagging. Inoltre, può risultare importante effettuare una normalizzazione del testo (es. stemming), sia allo scopo di ridurre le variazioni tra token simili, sia per migliorare la recall di sistemi di classificazione costruiti su uno spazio vettoriale.
Successivamente, possono venire applicati anche altri sistemi per migliorare la rappresentazione dell'informazione presente all'interno della matrice. In Lowe (2001)[38] il processo di costruzione per uno spazio vettoriale viene suddiviso in un processo a quattro fasi: calcolo delle frequenze, trasformazione delle frequenze grezze in pesi statistici, riduzione della dimensionalità e calcolo delle similarità (Turney e Pantel, 2010)[51].
Le prime due fasi della costruzione sono sufficientemente auto esplicative: il corpus viene scandito contando le frequenze di ogni elemento, e la struttura viene in seguito trasformata nella matrice di co-occorrenze.

L'uso di pesi statistici deriva ancora una volta da un'intuizione proveniente dalla teoria dell'informazione, secondo la quale un evento sorprendente ha un contenuto informativo più rilevante di un evento atteso (Shannon, 1948)[66]. La formalizzazione matematica più popolare di questa intuizione è la famiglia di misure *tf-idf* (text frequency x inverse document frequency) (Sparck Jones, 1972)[30], per le quali un evento ha un valore tanto più alto quanto alta è la frequenza in un particolare documento rispetto alla frequenza negli altri (es. una parola che appare frequentemente in un solo documento avrà un valore molto elevato).

Un alternativa è rappresentata dalla *PMI (Pointwise Mutual Information)* (Church e Hanks, 1989 [16]; Turney 2001 [73]).
La formula della PMI è la seguente:

$$PMI(w_1, w_2) = log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)} \tag{2.1}$$

La misura è calcolata come la stima probabilistica dell'occorrenza di una parola all'interno di un contesto. Se gli elementi sono statisticamente indipendenti il valore è 0, se invece esiste un'associazione significativa il valore è più elevato. Esiste anche il caso in cui una parola può essere completamente slegata dal contesto per cui il valore viene calcolato. In questo caso il valore ottenuto è inferiore a 0. Proprio per questo è stata proposta una variazione, detta Positive PMI (Niwa e Nitta, 1994)[45], per la quale i valori inferiori allo 0 vengono trasformati in 0 (indipendenza statistica). La misura è efficace sia per matrici termine-contesto che per matrici termine-documento(Pantel e Lin, 2002a[49]; Pantel e Lin, 2002b[50]). Tuttavia, un problema noto della PMI riguarda la sua tendenza a premiare eventi non frequenti. Sono quindi stati introdotti in letteratura numerosi fattori volti a ovviare a tale problema.

La matrice, dopo essere stata creata, necessita inoltre di essere ridotta di dimensionalità e sottoposta a un processo di smoothing. Questo si rende necessario poiché nella maggior parte dei casi la matrice risulta sparsa (cioè composta da un gran numero di elementi nulli).
Molte tecniche sono state proposte in letteratura per ovviare alle dimensioni considerevoli della matrice, e per ridurre i calcoli da eseguire pur mantenendo un'adeguata precisione. Lin (1998)[34] ha mostrato ad esempio come una semplice soglia applicata al valore di PMI degli elementi della matrice sia in grado di ridurre enormemente il numero di comparazioni da eseguire pur ritenendo una precisione sufficiente.
Sono state inoltre proposte numerose altre tecniche, tra le quali è necessario menzionare il *Truncated SVD (Singular Value Decomposition)*, proposto in Deerwester et al. (1990)[19], e utilizzato con successo per vari scopi, tra cui la scoperta di dimensioni latenti nei dati(Deerwester et al., 1990 [19]; Landauer e Dumais, 1990 [33]),

riduzione del rumore all'interno della matrice (Rapp, 2003)[61], co-occorrenze di alto ordine (Landauer e Dumais, 1997[33]) e infine riduzione della sparsità della matrice (Vozalis e Margaritis, 2003)[80].

Dopo aver creato la matrice e aggiustato i suoi elementi è fondamentale infine eseguire il calcolo della similarità tra gli elementi, scopo ultimo della creazione di spazi vettoriali.
Il metodo più comune per calcolare la similarità tra due vettori di una matrice risulta essere il calcolo del coseno dell'angolo sotteso a essi (Turney e Pantel, 2010)[51]. La formula per calcolare il coseno tra due vettori $\vec{a}$ e $\vec{b}$ è la seguente:

$$cos\ \theta = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}||\ ||\vec{b}||} \tag{2.2}$$

La misura è effettuata il calcolando il prodotto interno dei vettori, dopo che questi sono stati sottoposti a normalizzazione per ridurli a lunghezza unitaria. Questo avviene perché, nella misura del coseno, si ritiene la lunghezza dei vettori irrilevante rispetto invece all'angolo tra essi. Se si pensa ad una rappresentazione bidimensionale su un piano cartesiano ad esempio, due vettori sono tanto più simili quanto più l'angolo tra essi si riduce, senza considerare la lunghezza effettiva dei vettori. I valori assunti dal coseno possono spaziare tra -1 per un angolo di 180°, che rappresenta quindi vettori opposti, a 1, per un angolo di 0°, che rappresenta due vettori sovrapposti, che puntano nella stessa direzione. Il valore 0 per la misura del coseno rappresenta infine due vettori ortogonali, con quindi un angolo fra essi misurato 90°, come mostrato in figura.

Oltre al coseno sono state proposte numerose altre misure per calcolare la similarità, che tuttavia una volta normalizzati correttamente i vettori in generale non danno luogo a divergenze significative (Van Rijsbergen, 1979)[78]. Dal punto di vista geometrico sono molto popolari la *distanza Euclidea* e la *Manhattan distance*, mentre altri esempi possono essere il *coefficiente di Jaccard* e *Dice* (Manning et al., 2008)[40]

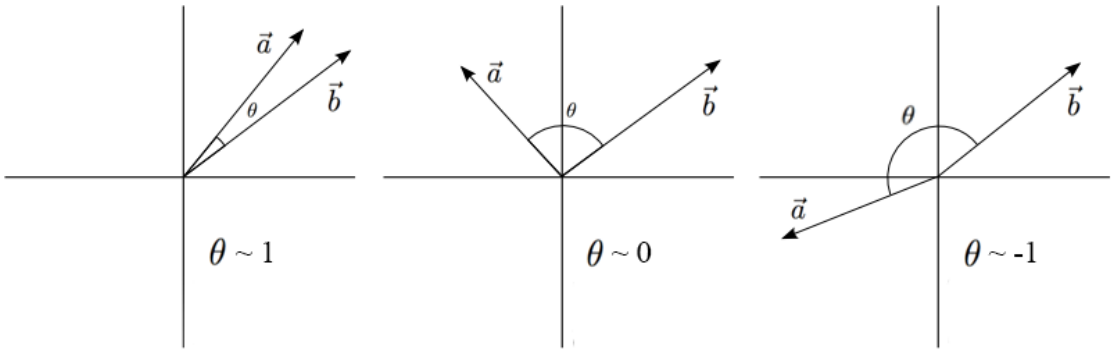

Figura 2.1: Misure del coseno dell'angolo per i 3 casi base di vettori simili, non correlati e opposti

Sicuramente nel panorama della letteratura scientifica riguardo l'analisi del linguaggio naturale i modelli a spazi vettoriali avranno una posizione prominente negli anni a venire, poiché in grado di realizzare in pratica l'idea che il significato delle parole sia strettamente connesso alla statistica del loro utilizzo (Turney e Pantel, 2010)[51]. Proprio per questa ragione, in Turney e Pantel (2010)[51], a cui si rimanda ancora una volta per una trattazione più completa riguardo l'argomento, si giunge alla conclusione che per realizzare tutto il potenziale dei computer, è necessario renderli in grado di comprendere la semantica del linguaggio naturale, e proprio i modelli a spazi vettoriali potrebbero essere parte della soluzione a questo problema.

2.4 Classificazione con Support Vector Machines

Le *SVM (Support Vector Machines)* stanno negli ultimi anni giocando un ruolo decisivo in vari campi scientifici, per la loro capacità di garantire tecniche che sono specialmente versate a ottenere risultati in maniera efficiente e con un buon livello qualitativo (Janmenjoy et al., 2015)[27]. Si tratta di macchine computazionali con un approccio supervisionato capaci di risolvere problemi di regressione, clustering e classificazione. In particolare quest'ultimo task, secondo (Janmenjoy et al., 2015)[27], risulta quello più esplorato in letteratura, con il 54% delle applicazioni analizzate dalla ricerca, e quello più interessante anche ai fini della presente tesi.

La prima proposta in ambito di SVM risale a Vapnik (1998)[79], nell'ambito della teoria dell'apprendimento statistico e della minimizzazione del rischio strutturale (Janmenjoy et al., 2015)[27]. Lo scopo di una SVM, soprattutto per quanto riguarda la classificazione, è quello di minimizzare il rischio connesso all'errore di classificazione, individuando i limiti di separazione migliore tra una serie di dati. Questi sono rappresentati attraverso vettori di feature, le quali vengono pesate dall'algoritmo in fase di training per garantire che i margini che separano classi differenti siano massimizzati nello spazio che rappresenta i dati.

Nella pratica questo viene ottenuto con una rappresentazione dei dati, basata sulle feature che li caratterizzano, all'interno di un piano. Una SVM è quindi un iperpiano con margine massimo poggiato sullo spazio, in grado quindi di classificare i dati separati da confini non lineari (cioè nella rappresentazione dei dati non esiste una linea di separazione univoca tra elementi di una classe e elementi dell'altra). Viene costruito tramite la previa individuazione di un set di iperpiani che separano le due, o più, classi dei punti all'interno dello spazio. Dopo aver costruito gli iperpiani, l'algoritmo dell'SVM individua i confini tra le classi in input e gli elementi in input definendone i confini (support vector) (Sivanandam et al., 2006)[69]. Dunque dato un set di esempi di training, etichettati con la classe positiva o negativa, compito dell'SVM è quello di individuare l'iperpiano che li divide avente il margine massimo. Se tale iperpiano non esiste, ovvero se le classi per i dati non sono linearmente separabili, l'algoritmo sceglie comunque l'iperpiano che divide gli esempi in maniera più netta possibile, sempre massimizzando il margine tra gli esempi più vicini (Janmenjoy et al., 2015)[27]. Il problema della classificazione quindi diventa un problema di ottimizzazione del valore di margine, che può altrimenti essere visto come un problema di minimizzazione del peso W nell'equazione dell'iperpiano che rappresenta l'SVM: $W * p + b = 0$, dove p è l'esempio di training e b la propensione dell'iperpiano.
Come esempio, nelle figure sono rappresentate sia la trasposizione dallo spazio di input allo spazio delle feature, con l'iperpiano che divide le due classi, sia una rappresentazione piana di alcuni iperpiani tra i quali solo uno massimizza il margine tra

gli elementi delle due classi.

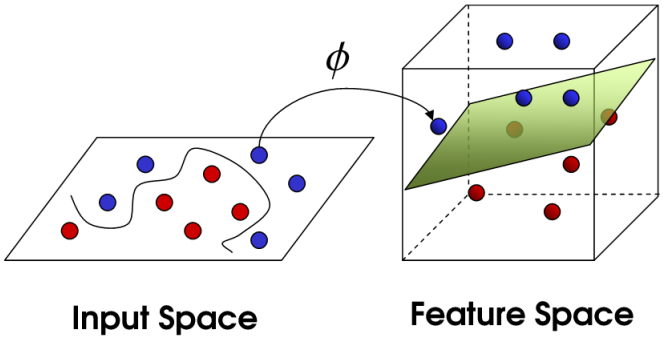

Figura 2.2: Da spazio di input a spazio di feature

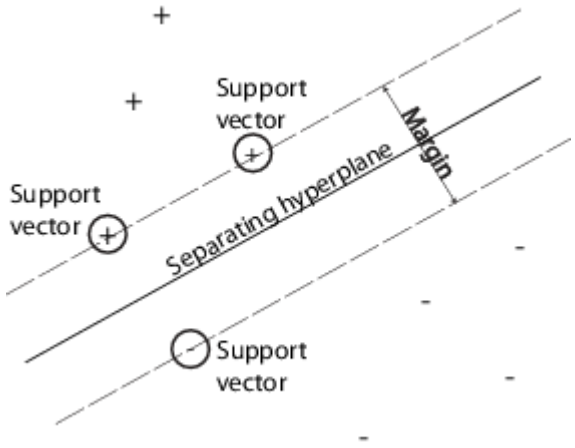

Figura 2.3: Esempio di iperpiano che divide lo spazio massimizzando il margine

Le support vector machines in letteratura sono state utilizzate con successo in un gran numero di casi, spesso migliorando decisamente prestazioni di algoritmi e processi alternativi. Inoltre, grazie alla loro capacità di generalizzazione attraverso i vettori di feature, l'utilizzo non è limitato a un campo di studi specifico. Sono difatti numerosi gli esempi di ricerche estremamente fruttuose nell'ambito dell'utilizzo di SVM. Di seguito sono riportati alcuni esempi in merito nelle varie discipline scientifiche.

In Zanaty (2012)[87] è presente una comparazione con altri tipi di modelli per la

classificazione dei dati, in particolare rispetto a un classificatore *Multi Layer Perceptron (MLP)*, mostrando come l'implementazione di SVM, soprattutto per dataset molto ampi, risulti più rapido. Wan et al. (2012)[82] invece hanno proposto un sistema ibrido per la classificazione di documenti testuali che potesse integrare un approccio alla classificazione di tipo *K-nearest neighbor (KNN)* con l'algoritmo di addestramento SVM, mostrando ottimi risultati soprattutto in contrasto con il semplice utilizzo di K-NN. Zhang et al. (2008)[81] hanno implementato l'estrazione di multi-word dal testo basata sulla struttura sintattica delle frasi multi-word nominali. Attraverso una serie di modelli SVM si è cercato di stabilire l'impatto di differenti funzioni kernel, ovvero di misure di similarità, sulle performance di classificazione. Gli esempi sopra citati provengono da Janmenjoy et al. (2015)[27], al quale si rimanda per una trattazione più ampia.
Infine, è necessario citare Passaro et al. (2014)[53], di cui si tratterà nel paragrafo seguente 2.4.1, in cui un classificatore SVM è stato utilizzato per definire la polarità positiva o negativa di tweet presenti all'interno di un dataset.

Risulta dunque chiaro, come evidenziato in Janmenjoy et al. (2015)[27], quanto i metodi e modelli basati su SVM possano essere sviluppati e utilizzati per un gran numero di aree applicative differenti, tra cui medicina, ingegneria, problemi di classificazione e soprattutto, ai fini della presente ricerca, per elaborazione del linguaggio naturale e in particolare sentiment analisys e classificazione della polarità di documenti.

Ciononostante è importante anche denotare alcuni limiti e potenziali aree di sviluppo futuro delle SVM in generale. Alcuni dei più importanti possono essere la scelta del kernel da utilizzare per un determinato problema, oltre alla scelta di parametri di buona qualità per esso, la convergenza lenta soprattutto in fase di test, e soprattutto la grande quantità di memoria necessaria all'implementazione del modello (Janmenjoy et al., 2015)[27].

Le SVM dunque risultano allo stato attuale essere un campo di studio estremamente fruttuoso,anche e soprattuto, come già precisato, per quanto concerne lo studio di tecniche di elaborazione del linguaggio naturale ed estrazione della conoscenza, quindi di grande interesse per la presente ricerca e per il lavoro da cui questa prende le mosse, presentata in Passaro et al. (2014)[53].

2.4.1 Evalita 2014 - SentiPolC

Proprio da quest'ultima ricerca è stato garantito un punto di partenza solido ed interessante per il presente progetto, in particolare per quanto riguarda l'implementazione del classificatore volto a individuare la polarità dei tweet. DIifatti in Passaro et al. (2014)[53] è stato utilizzato un classificatore basato su SVM per la classificazione. L'esperimento è stato svolto nell'ambito della competizione Evalita 2014, in particolare per il task di *sentiment polarity classification* (SentiPolC). Come evidenziato nel report, l'utilizzo di un sistema basato su SVM è stato prescelto per le migliori prestazioni evidenziate in task analoghi (Passaro et al., 2014)[53]. Il sistema scelto è stato basato su su un classificatore SVM con una implementazione di kernel lineare disponibile in Weka (Witten et al., 2011)[84], addestrato attraverso l'algoritmo di Sequential Minimal Optimization (SMO) introdotto da Platt (1998)[58].

Il classificatore è stato addestrato inizialmente su due set di feature differenti, poi combinati per produrre il modello finale. Il linguaggio utilizzato su Twitter dagli utenti è estremamente peculiare, spesso influenzato dalla frequenza di punteggiatura creativa, emoticon, slang, terminologia specifica, abbreviazioni, link e hashtag, concentrati nei 140 caratteri di testo a disposizione (Passaro et al., 2014)[53]. Proprio per questo motivo i ricercatori hanno dapprima approntato un modello che andasse a utilizzare come feature proprio questo genere di caratteristiche del corpus a disposizione, oltre a un modello puramente lessicale dove al contrario non è tenuto conto di esse. Il modello non lessicale è stato appunto utilizzato come baseline principale per lo sviluppo del modello creato ai fini della presente tesi.

Per quanto riguarda il modello non lessicale, chiamato altrimenti Shallow, il set di feature utilizzato per la classificazione si compone di:

Feature morfologiche come ad esempio numero di frasi, numero di token linguistici, proporzione di parole piene, numero di token per Part-Of-Speech.

Emoticon Attraverso l'utilizzo della risorsa proprietaria EmoLex, che contiene le emoticon più popolari associando ciascuna di esse a un punteggio di polarità.

Links Classificazione attraverso espressioni regolari dei link presenti, al fine di individuare video, immagini, e social network.

Enfasi Numero di token enfatizzati, ovvero contenenti caratteri ripetuti non presenti nella forma standard.

Punteggiatura creativa Sequenze di caratteri contigui di punteggiatura, come ad esempio "!!!" o "??!!".

Citazioni Numero di citazioni presenti nel tweet.

Al contrario, il modello lessicale utilizza una serie di risorse lessicali per individuare la polarità di ciascuno dei termini presenti.
Innanzitutto è stato utilizzato il lessico emotivo prodotto per ItEM (Passaro et al., 2015)[54], una risorsa contenente una serie di parole altamente emotive, utilizzato per individuare il numero di token altamente emotivi all'interno del tweet. Inoltre, è stato utilizzato un lessico volto ad individuare le più comuni parolacce dell'italiano. Successivamente. è stato implementato anche un metodo semplificato per il riconoscimento delle negazioni, a partire ancora una volta da un inventario di lemmi e pattern negativi, estratto da Renzi et al. (2001)[62].
Infine, risulta necessario spendere qualche parola in più riguardo la risorsa lessicale vera e propria utilizzata per la creazione del sistema CoLingLab, in quanto oggetto di dibattito nei capitoli successivi. Si tratta di *Sentix* (Sentiment Italian Lexicon) (Basile e Nissim, 2013)[6], un lessico contenente circa 60000 lemmi italiani annotati con valori di polarità e intensità. Il lessico è a sua volta stato creato allineando

alcune risorse già disponibili, tra cui WordNet, MultiWordNet e SentiWordNet. I valori di polarità e intensità sono stati ottenuti da SentiWordNet. Questi sono stati utilizzati per dividere i lemmi in cinque classi differenti, da altamente positivi ad altamente negativi. Risulta importante notare come la risorsa lessicale in questione abbia richiesto uno sforzo considerevole da parte dei ricercatori per essere portata a termine, in quanto frutto dell'interpolazione di varie altre risorse oltre che del necessario sviluppo di nuove misure per la definizione dei valori di polarità e intensità di tutti i lemmi presenti nella risorsa lessicale.

Sebbene il modello migliore sia risultato la combinazione dei due modelli illustrati, lessicale e non lessicale, quest'ultimo ottiene risultati migliori per quanto riguarda la Precision, ed è in grado di riconoscere più accuratamente i tweet positivi (Passaro et al., 2014)[53]. Se confrontato inoltre con il modello puramente lessicale, risulta evidente come le prestazioni siano generalmente migliori. Sebbene quindi sia stato mostrato come possano essere ottenuti risultati significativamente migliori con l'aggiunta di feature di tipo lessicale, queste hanno più difficoltà a ottenere migliori prestazioni se non supportate da feature non lessicali.

Il modello sviluppato si è classificato in terza posizione nella competizione Evalita 2014, task SentiPolC.
Nelle tabelle presentate di seguito sono riportati i risultati ottenuti dai tre modelli di classificazione sviluppati. Per una trattazione più completa si rimanda a (Passaro et al., 2014)[53].

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7979	0.7806	0.789
POS	1	0.581	0.4109	0.4814
POS task		0.6893	0.5957	0.6352
NEG	0	0.6923	0.6701	0.681
NEG	1	0.6384	0.5201	0.5732
NEG task		0.6654	0.5951	0.6271
GLOBAL		0.6774	0.5954	0.6312

Tabella 2.1: Risultati sistema CoLingLab

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7599	0.7755	0.7676
POS	1	0.4913	0.2981	0.371
POS task		0.6256	0.5368	0.5693
NEG	0	0.66	0.6861	0.6728
NEG	1	0.6218	0.4522	0.5237
NEG task		0.6409	0.5692	0.5983
GLOBAL		0.6333	0.553	0.5838

Tabella 2.2: Risultati sistema CoLingLab, Lexical Model

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7578	0.8679	0.8092
POS	1	0.7184	0.2205	0.3374
POS task		0.7381	0.5442	0.5733
NEG	0	0.7369	0.5174	0.608
NEG	1	0.5778	0.6582	0.6154
NEG task		0.6574	0.5878	0.6117
GLOBAL		0.6978	0.566	0.5925

Tabella 2.3: Risultati sistema CoLingLab, Shallow Model

Capitolo 3

Creazione del corpus

Nel presente capitolo sarà analizzata la fase preliminare di creazione del corpus. La creazione della risorsa è avvenuta attraverso una serie di fasi distinte. In primo luogo, oltre alla scelta del tipo di dati da considerare, sono stati analizzati gli eventuali problemi posti dall'utilizzo delle API fornite da Facebook. In seguito si è proceduto all'effettivo recupero dei dati ricavati dal social network attraverso l'utilizzo di un crawler appositamente creato per lo scopo in linguaggio Python. Infine, una fase di post-processing dei dati testuali grezzi ottenuti è stata necessaria, sia per limitarli al lasso temporale scelto come obiettivo, sia per ripulirli da eventuali impurità e renderli utilizzabili da parte degli strumenti per le analisi successive.

3.1 Pagine Facebook e Graph API

3.1.1 Pagine

Come già affermato nel capitolo introduttivo, l'aumento del bacino di utenza delle reti sociali avvenuto negli ultimi anni ha condotto l'informazione di massa sempre più a spostarsi da siti specializzati a piattaforme più generiche, promuovendo di fatto l'interazione degli utenti, ed andando a creare come già menzionato un dibattito vero e proprio, entro cui gli utenti oltre a commentare la notizia in se si trovano a discutere fra loro delle proprie opinioni.

Proprio a questa tipologia di dati ci si è interessati per la presente ricerca, al fine di ottenere un corpus con caratteristiche fortemente emotive.

Per la creazione del corpus è stata quindi effettuata la scelta di considerare le pagine Facebook ufficiali di alcune delle maggiori testate giornalistiche italiane, sia cartacee che web. Si è cercato di individuare testate con una sufficiente diversità sia contenutistica che di visione, soprattutto in termini politici, in modo da ottenere dati non sbilanciati verso una tematica particolare, ma dal contenuto emotivo vario per i diversi argomenti di volta in volta oggetto di dibattito.
Il social network, come già accennato, permette agli utenti, quali possono essere le pagine, di pubblicare stati, link, foto e video sulla propria bacheca, una vera e propria *timeline* sequenziale di elementi riguardanti la propria vita.
Si possono suddividere le tipologie di utenti in due macro-categorie generali: *persone* e *pagine*. Le persone rappresentano utenti singoli, persone fisiche appunto, e l'interazione con esse è solitamente bidirezionale. Per poterne osservare la timeline completa, ovvero i post pubblicati, è necessario inviare una *richiesta di amicizia*, che deve essere approvata da chi la riceve. Le pagine invece solitamente sono utilizzate per rappresentare organizzazioni, personaggi pubblici, imprese o, come nel caso specifico, redazioni di quotidiani. La relazione è in questo caso unidirezionale, rappresentata dal *like* che un utente può o meno garantire alla pagina, e che semplicemente permette di ricevere gli aggiornamenti dalla determinata pagina nel proprio *news feed*, ovvero la pagina principale di Facebook, in cui ogni utente riceve gli aggiornamenti sotto forma di post da parte dei propri amici e delle pagine che segue (cioè le pagine a cui ha apposto il proprio like). Va notato comunque che, a meno di specifiche direttive dei gestori della pagina, gli elementi pubblicati da esse sono comunque, solitamente, pubblici e visibili a tutti nella timeline delle stesse, senza la necessità del like.
A differenza di un utente singolo, ovvero una persona, che può essere più incline a pubblicare stati, foto, video e link riguardanti la propria vita personale, le proprie esperienze o opinioni, sulle pagine, ed in particolare le pagine dei quotidiani, la nor-

ma, da cui molto raramente ci si allontana, è quella di pubblicare link di notizie che rimandano al sito ufficiale della testata[1], dove è presente l'articolo vero e proprio. Ci si trova quindi davanti a link che mostrano il titolo della notizia, molto spesso preceduti da una breve didascalia a commento della notizia stessa.
È chiaro come l'effettivo contenuto emotivo dei post possa in questo caso non avere un grande impatto, se non quello di suscitare reazioni negli utenti che leggono il titolo della notizia. Risulta infatti decisamente più interessante la sezione dei commenti, disponibili per ogni post, e nei quali come già affermato gli utenti si scambiano opinioni riguardanti la notizia pubblicata. La gerarchia prevede quindi un livello più alto costituito dai *post*, a cui fanno capo una serie di *commenti*, ciascuno dei quali è relativo allo specifico post. Esiste inoltre un ulteriore livello gerarchico, rappresentato dalle *risposte*, che possono essere considerate, dal punto di vista concettuale ma anche pratico, come dei commenti riferiti ad un particolare commento. L'esempio della gerarchia presente nei post di Facebook è mostrato in figura. Ogni utente può quindi pubblicare sulla propria timeline una serie di post. Ogni post può avere un certo numero di commenti, ed ogni commento può avere un certo numero di risposte. Spesso quindi le risposte, pur rimanendo nell'ambito informativo del post sotto il quale vengono scritte, sono direttamente riferite al singolo commento cui fanno capo.

Si è deciso, per la presente ricerca, di utilizzare i dati di post, commenti e risposte ricavati analizzando le bacheche delle seguenti pagine di testate:

- La Repubblica
- L'Avvenire
- Il Giornale

[1]Va precisato che di recente Facebook ha introdotto la possibilità, soprattutto rivolta ai giornali, di pubblicare la notizia nella sua interezza direttamente sul social. Tuttavia questa pratica non è ancora utilizzata da nessuna delle testate prese in considerazione per la presente ricerca nell'arco di tempo interessato

Figura 3.1: Esempio di post, commenti e risposte su Facebook

- Libero
- Il Fatto Quotidiano
- Rainews24
- Il Corriere Della Sera
- Huffington Post Italia

Si è deciso di restringere l'arco temporale considerato per la creazione della risorsa all'intero anno 2015, a partire dal 1 gennaio alle ore 00:00:01 fino al 31 dicembre alle ore 23:59:59. La scelta è stata dettata sia da ragioni di coerenza, in modo da avere a disposizione lo stesso arco temporale per tutte le testate, sia da ragioni computazionali, vista la elevata quantità di dati da analizzare.

3.1.2 Graph API

Il metodo primario di crawling e recupero dati dalla piattaforma Facebook è costituito dalla *Graph API*, una API di basso livello basata su HTTP, che può essere utilizzata per richiedere dati, pubblicare nuovi elementi e in generale gestire ciò che un'applicazione costruita per Facebook possa richiedere (Facebook Graph API, 2016)[22].
Il servizio di API [2] permette infatti di navigare attraverso il grafo del social network. La struttura del grafo è organizzata come segue:

Nodi Le entità della rete, quali ed esempio Utenti, Pagine, Foto e Commenti.

Archi Connessioni tra i nodi della rete. Ad esempio, le foto di una determinata pagina.

Campi Informazioni riguardanti una determinata entità. Possono rappresentare ad esempio la data del compleanno di un utente o il numero di likes ottenuti da un post.

[2] `https://developers.facebook.com/docs/graph-api`

La navigazione del grafo viene effettuato tramite chiamate di tipo HTTP, che possono quindi essere implementate tramite un qualsiasi linguaggio di programmazione che abbia la possibilità di supportarle. Nel presente caso si è scelto, per la sua semplicità ed immediatezza, di utilizzare Python.
L'API mette a disposizione numerosi strumenti per utilizzare il grafo. In particolare, per la presente ricerca è risultato importante avere la possibilità di utilizzare una formattazione dei dati richiesti basata sulla data di pubblicazione, potendo effettuare sia richieste del tipo "fino a <data>" o "da <data> in poi", sia concatenandole insieme.
Va comunque notato come lo strumento sia caratterizzato da limitazioni anche piuttosto pesanti per gli utenti, soprattutto in un ottica come quella del crawling. Difatti, come già accennato precedentemente, solamente i dati pubblici sono accessibili senza richiedere alcun consenso da parte degli utenti. Inoltre, per ogni applicazione di recupero dati, sono poste delle limitazioni riguardanti il numero di chiamate HTTP consecutive possibili un un determinato arco di tempo. Questo rende difficile recuperare un gran numero di elementi consecutivamente prima di venire limitati o temporaneamente bloccati, poiché viene considerato ogni elemento (post, commento o risposta) come una singola chiamata. Anche per questo motivo, durante la fase di raccolta dei dati ci si è trovati davanti a prestazioni del crawler inconsistenti, dipendenti probabilmente oltre che dalle limitazioni già citate, anche dalla quantità di traffico dati che al momento veniva generato dall'utenza di Facebook. Un ultima limitazione degna di nota si è rivelata l'impossibilità di effettuare ricerche tramite keyword, presente in versioni più vecchie dell'API ma non più disponibile al momento in cui la presente risorsa è stata generata, se non per partner specifici scelti da Facebook.

3.2 Crawler

Come già precisato, il linguaggio scelto per creare il crawler è stato Python.

L'API di Facebook richiede, per effettuare le chiamate al server, di un *token* di accesso. Il token è ottenibile sia come utente iscritto a Facebook, sia come applicazione. La limitazione del codice utente è data da una durata estremamente limitata della validità dello stesso, di 60 minuti. Per questo è stato deciso di creare un'applicazione di appoggio, per la quale il codice di accesso rimane valido, senza necessità di rinnovo, per 60 giorni, un tempo più che sufficiente per raccogliere la risorsa.

Si è deciso di utilizzare, per il recupero del corpus, la possibilità garantita dall'API di ottenere una *time-based pagination* delle informazioni ottenute. Questo ha permesso di recuperare i dati in ordine cronologico inverso, dal post più recente fino al più remoto. Alla fine di ogni ciclo di esecuzione è stato salvato il valore del post più vecchio recuperato per ogni utente, ovvero ogni pagina, sotto forma di *timestamp unix*. Questa è costituita da un valore intero incrementale in grado di rappresentare una qualsiasi data, a partire dal 1 gennaio 1970, fino al livello di dettaglio dei secondo. Chiaramente ogni secondo in più è rappresentato da un incremento di 1 rispetto al valore precedente della timestamp. Oltre a risultare comodo dal punto di vista computazionale per la possibilità di effettuare tutte le operazioni possibili sugli interi (in particolare maggiore e minore, per individuare date precedenti o successive tra loro), risulta essere anche il formato utilizzato dalla Graph API per restituire le date di ogni elemento, rendendo non necessario alcun tipo di conversione in fase di raccolta dati.
La lista degli utenti rilevanti per la ricerca e delle relative timestamp più lontane nel tempo è stata gestita in un file separato.

La parte centrale dell'algoritmo è strutturata in modo da recuperare il nome utente e relativa timestamp più vecchia, ed effettuare una serie di chiamate per ottenere i dati esistenti fino a quella data. Ciò è ottenuto utilizzando il metodo *until* dell'API. Di seguito è riportato l'esempio di una chiamata HTTP, in cui vengono richiesti i post più vecchi del 13 luglio 2015 alle ore 11 esatte per la pagina di Repubblica:

```
GET graph.facebook.com
  /REPUBBLICA/feed?=posts&until=1436778000
```

Il codice utilizzato nell'algoritmo risulta essere leggermente differente, in quanto la libreria utilizzata per effettuare la stessa operazione in Python richiede una sintassi diversa. Inoltre, la richiesta effettuata tramite lo script risulta essere molto più articolata. Anche in questo caso è stata sfruttata una proprietà della Graph API, che consente di effettuare, utilizzando una sola riga di codice, una serie di richieste annidate. Nel presente caso, ad esempio, è stato necessario recuperare i post, i commenti relativi per ognuno di essi, e per ognuno dei commenti eventuali risposte. Chiaramente per ogni elemento sono state richieste informazioni addizionali, quali ad esempio il numero di *likes*, sulle quali si tornerà in seguito. Si riporta un altro piccolo esempio illustrativo del codice utilizzabile in Python per effettuare la richiesta, sempre per la pagina di Repubblica, e sempre per post, commenti e risposte più vecchi del 13 luglio 2015 alle ore 11:

```
until = '.until(1436778000)'
args = {'fields' : 'posts'+until+'(1436778000){message,
    story, created_time, type, name, likes.summary(true),
    shares, comments.summary(true){message,
    likes.summary(true),comments.summary(true){message,
    likes.summary(true),comments.summary(true),
    created_time}, created_time}}', }
graph = facebook.GraphAPI(access_token=access_token)
profile = graph.get_object('REPUBBLICA')
retrieved_data = graph.get_object(profile['id'], **args)
```

Come si può osservare, la variabile *args* rappresenta i campi da richiedere alla Graph API.

Come già accennato, si tratta di un esempio, volto soltanto a rendere comprensibile la struttura delle richieste effettuate. Il processo completo chiaramente integra una

gestione di eventuali errori ed eccezioni, oltre ad automatizzare il processo di acquisizione del nome utente e della timestamp attuale per ogni utente. Per il codice completo del crawler si rimanda all'appendice.

Per quanto concerne i dati addizionali, ovvero i campi, effettivamente recuperati tramite il crawler, si è scelto di operare una selezione, chiarita nell'esempio dagli argomenti della chiamata.
Per ogni post si è scelto di salvare e conservare le seguenti informazioni:

message : il testo del post in questione.

story : informazioni aggiuntive riguardanti il post. Ad esempio, potrebbe essere l'aggiunta di una foto, o un particolare tag riferito ad un altro utente.

created time : la data di creazione del post, rappresentata come già spegato tramite una timestamp unix, riferita all'orario locale dell'utente.

type : specifica se è un post o un commento (commento o risposta).

name : nel caso sia pubblicato un link, ne contiene il nome del titolo.

likes.summary(true) : numero di likes ottenuti dal post.

shares : numero di condivisioni ottenute dal post.

comments.summary(true) : numero di commenti, oltre ovviamente ai commenti stessi.

Per quanto concerne invece i commenti e le risposte, la sintassi della richiesta annidata è identica, come evidenziato nell'esempio, ma ci si è limitati a considerare il testo, la data di creazione, il numero di like ed eventuali risposte.
Va infine considerato come, di default, per ogni post, commento e risposta sia anche restituito l'*ID* univoco a esso associato.

Una volta ottenuti i dati tramite la chiamata HTTP GET, questi sono stati analizzati per poter essere scritti su file.
Ogni post recuperato è stato quindi sequenzialmente esaminato da apposite funzioni per recuperare le informazioni in esso contenute e, ricorsivamente, così è stato anche ogni commento e risposta riferiti ad esso.
Per facilitare le analisi successive i post sono stati scritti su file con una precisa struttura, di cui è riportato di seguito un esempio per post, commento e risposta:

```
<doc user="corrieredellasera"
    id="284515247529_10153481614722530" type="post"
    parent_post="" parent_comment="" date="1445520489"
    location="" likes="531" comments="21" shares="221">

Invecchiamento, le abitudini di vita e di alimentazione
    hanno un ruolo determinante.

</doc>

<doc user="corrieredellasera"
    id="10153481614722530_10153481616402530"
    type="comment"
    parent_post="284515247529_10153481614722530"
    parent_comment="" date="1445520553" likes="3"
    comments="0">

oggi mi sento estremamente romantico....
con questa frase vi auguro un buon pomeriggio .

</doc>

<doc user="corrieredellasera"
    id="10153481614722530_10153481711682530"
```

```
type="comment"
parent_post="284515247529_10153481614722530"
parent_comment="10153481614722530_10153481656132530"
date="1445524127" likes="1" comments="0">

Solo che nasce vecchia e continua a invecchiare

</doc>
```

Come si può notare, l'utente considerato per commenti e risposte è stato l'autore del post, ovvero la pagina interessata. Oltre alle informazioni precedentemente considerate, sono stati aggiunti anche due campi che contengono, se si tratta di un commento, l'informazione relativa al post padre, ovvero quello sotto al quale il commento è stato espresso, e se si tratta di una risposta anche l'informazione riguardante il commento genitore. In caso contrario i campi sono stati lasciati vuoti.
Infine, insieme al corpus è stato prodotto un file di supporto contenente, per ogni elemento, le informazioni relative a post padre, commento genitore e eventuale storia. Chiaramente, per i post è stato salvato solamente l'ID, per i commenti l'ID e il post padre, mentre per le risposte tutti e tre i campi.
L'algoritmo è stato quindi avviato manualmente più volte in sequenza, in modo da ottenere la collezione di dati prefissata. Il processo ha richiesto circa tre settimane. Va chiarito come questa prestazione, per una quantità simile di dati, possa essere migliorata ulteriormente introducendo un tipo di automazione che permetta di eseguire più volte lo script sequenzialmente, con pause prefissate tra una run e l'altra così da non incorrere nelle limitazioni imposte dall'API. Si è scelto tuttavia di procedere manualmente, per semplicità ma anche e soprattutto per avere un controllo più diretto sulla creazione della risorsa e sulla gestione di eventuali errori. Questi difatti possono essere di vario tipo, spesso causati dai limiti prefissati da Facebook, e non sempre ben documentati nelle risorse messe a disposizione degli utenti dalla piattaforma.

3.3 Post processing e risultati

Dopo aver raccolto i dati per l'arco temporale richiesto, sono state effettuate alcune semplici operazioni di post processing volte a rendere la risorsa adatta alle successive elaborazioni.

La fase iniziale di post processing dei dati grezzi è iniziata con la sostituzione dei caratteri rappresentanti delle url con il simbolo "URL". Inoltre, il testo è stato tokenizzato e in seguito taggato con le Part-Of-Speech corrette. Il processo è stato effettuato utilizzando la pipeline per il natural language processing del Labratorio di Linguistica Computazionale dell'Università di Pisa.

Sono state in seguito eseguite alcune semplici statistiche sul testo volte a individuare la dimensione del corpus collezionato. Le statistiche, ottenute tramite un semplice script python, sono riportate in tabella.

Page	Post	Comment	Reply	Token
Repubblica	24829	3248123	1285877	95326364
Avvenire	6726	62684	22414	2593732
Il Giornale	44315	2971825	481470	63799437
Libero	29400	2151188	255658	40841255
Il Fatto Quotidiano	25217	3628701	1246396	98279833
Rainews24	32982	267562	69290	7685122
Huffington Post It	18246	1090003	443793	32340049
Il Corriere Della Sera	37015	2709861	807090	63704038
OVERALL	218730	16129947	4611988	404569830

Tabella 3.1: Numero di post, commenti e risposte per ogni pagina del corpus

Come si può notare, il corpus risulta di dimensioni più che consistenti per poter effettuare analisi distribuzionali, con un numero totale di token di circa 404 milioni. Per quanto riguarda le differenze tra i vari quotidiani si nota come, al di là della pagina di Avvenire, per il quale sono stati ricavati dati piuttosto limitati, probabilmente dovuti a un'attività non troppo assidua sul social network, le altre si attestano in

generale su valori abbastanza simili. La media del numero totale di post per ogni pagina è di 27341, mentre per quanto riguarda commenti e risposte è rispettivamente di 2016243 e 576498. Si può inoltre osservare come, generalmente, siano presenti un numero di commenti molto più elevato del numero di risposte, e che questi, chiaramente, superino di diversi ordini di grandezza il numero di post pubblicati da ogni pagina. Difatti, per ogni post pubblicato, soprattutto per quanto riguarda le pagine più attive e con un maggior numero di like, quali ad esempio Repubblica e Corriere Della Sera, le uniche a superare i 2 milioni di utenti, l'ordine di grandezza dei commenti è normalmente intorno alle migliaia. Va notato inoltre come anche per pagine di giornali che si potrebbero definire controversi, ovvero generalmente esponenti di posizioni politiche piuttosto estreme e radicali, il numero di commenti per ogni post è in generale relativamente elevato rispetto al numero di like effettivi alla pagina, come nel caso della pagina gestita dalla redazione di Libero.

3.4 Statistiche aggiuntive

Avendo a disposizione un così grande numero di post e commenti, è risultato interessante, prima di procedere, effettuare alcune semplici analisi volte anche a comprendere le dinamiche del social network preso in considerazione per la risorsa. In particolare, ci si è voluti soffermare sulla distribuzione dei commenti in relazione al post per cui sono stati prodotti e sulla distribuzione nella pubblicazione di elementi all'interno delle varie fasce orarie della giornata, sia per post che per commenti.
Si fa notare come nei casi che si andranno ad analizzare i commenti e le risposte sono trattati egualmente, e considerati tutti come commenti. Si è deciso di procedere in questo modo sia per semplificare le operazioni di analisi, sia perché, all'atto pratico, per questo tipo di analisi il fatto che si tratti di un commento o una risposta non porta informazione aggiuntiva.

3.4.1 Distribuzione di commenti per ogni post

Come già affermato, la prima analisi effettuata si è soffermata sul numero di commenti prodotti per ognuno dei post scaricati.
Trattandosi di pagine di quotidiani, che riportano notizie di attualità e sono seguiti da migliaia di persone, ci si aspetta chiaramente che il numero di commenti possa superare le migliaia. Difatti, il post con più commenti ne conta addirittura 341834, mentre in media ogni post ha 533 commenti, con una deviazione standard $\sigma = 1814.81$.
Per quanto invece concerne l'effettiva distribuzione dei commenti per ogni post all'interno del corpus, ci si è chiesti ovviamente se questi seguissero una legge di scala o di potenza. È stato quindi disegnato un grafico volto a rappresentare il numero di commenti relativamente al numero di post per i quali sono stati prodotti. La rappresentazione, per essere più leggibile, è stata ovviamente presentata in doppia scala logaritmica.

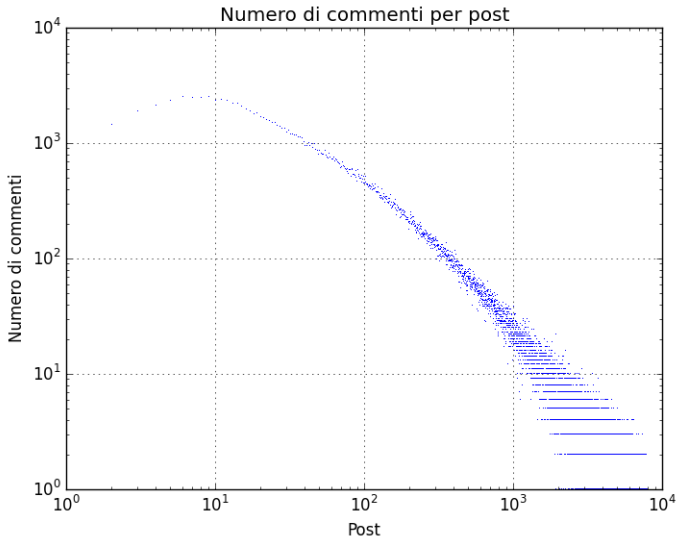

Figura 3.2: Numero di Post per Numero di commenti su Facebook

Come si può notare dalla figura, sembra abbastanza chiaro come i dati presentino chiaramente una legge di scala. In particolare la distribuzione risulta essere molto

simile a quanto teorizzato da Zipf (1949)[89] per quanto riguarda la distribuzione di frequenza delle parole di un testo scritto, e prima di lui da Auerbach (1913)[2] per quanto riguarda la distribuzione delle città rispetto al numero dei loro abitanti. La legge di Zipf infatti è una legge empirica che descrive la frequenza di un evento P_i facente parte di un insieme, in funzione del rango i nell'ordinamento decrescente rispetto alla frequenza stessa di tale evento.
Ciò diventa ancor più evidente se si osserva Figura 3.3, dove la scala del numero di commenti e dei post è stata adeguata per non considerare eventuali dati più sparsi agli estremi della distribuzione.

Si osserva chiaramente come siano presenti pochi post aventi un gran numero di

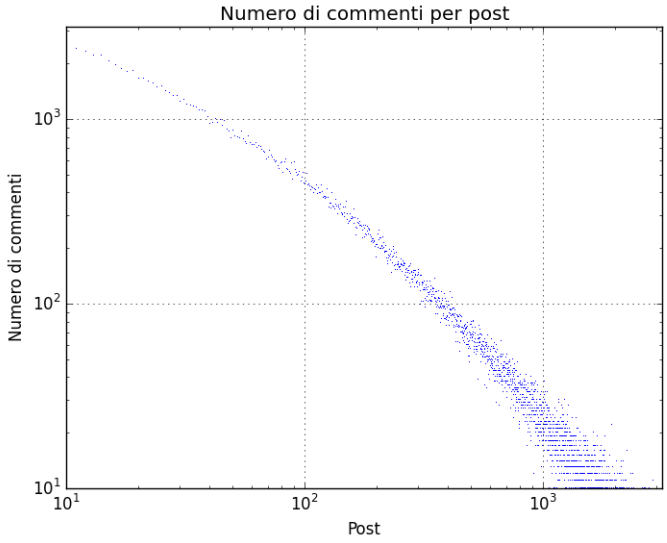

Figura 3.3: Numero di commenti per post, con scale adeguate alla distribuzione

commenti, mentre i dati riguardanti il numero di post diventano più densi mano a mano che il numero di commenti diventa sempre più basso.

Va ovviamente osservato come in questo caso si sia stato preso in considerazione un tipo di post particolare, ovvero appartenente a pagine con un gran numero di utenti attivi e pronti a commentare.Tuttavia è chiaro anche come, nonostante appunto siano stati considerati dati che tendenzialmente possono avere tanti commenti, la legge di scala risulta chiaramente rispettata.

3.4.2 Distribuzione oraria di post e commenti

Successivamente al calcolo del numero di commenti per ogni post, ci si è chiesti quando, nell'orizzonte temporale, vengano prodotti il maggior numero di post e commenti all'interno di Facebook. Questo tipo di dato, oltre che dal punto di vista analitico, può essere molto interessante anche da un punto di vista strettamente commerciale. Difatti, conoscere i momenti di maggiore attività per gli utenti del social network può portare a una migliore gestione del contenuto, come ad esempio la possibilità di effettuare una programmazione accurata riguardo la pubblicazione dei post più importanti, es. le notizie più rilevanti della giornata, in orari di punta nei quali ci sono maggiori possibilità di ottenere maggior visibilità e riscontro da parte della rete.

Si è quindi deciso di visualizzare graficamente la quantità di contenuti di tipo post e di tipo commento in base al loro orario di pubblicazione.
Per fare ciò, sono stati considerati tutti i post e tutti i commenti presenti nel corpus. Per ognuno di essi è stato salvato l'orario di pubblicazione, e successivamente questi sono stati aggregati proprio in base all'ora in cui sono stati pubblicati.

Per quanto riguarda il numero di post per ogni ora, i risultati sono mostrati in Figura 3.4.

Come si può notare dal grafico, è presente una distribuzione alquanto particolare. Sono prodotti un maggior numero di post nelle ore centrali della giornata, mentre durante la notte il numero cala drasticamente. Il picco massimo si può osservare intorno alle ore 11, seguito da un leggero ma evidente calo durante le ore del pranzo, per poi ritornare a elevarsi in frequenza durante le prime ore del pomeriggio. È interessante notare tuttavia come la distribuzione sia leggermente differente per quanto riguarda le prime ore del giorno rispetto alle ultime ore del pomeriggio e della sera, i due momenti in cui è evidente rispettivamente una crescita e una decrescita nel numero di post pubblicati. Mentre la crescita durante le prime ore del giorno, in particolare tra le 04 e le 07 del mattino, è rappresentata da una curva molto ripida, ciò

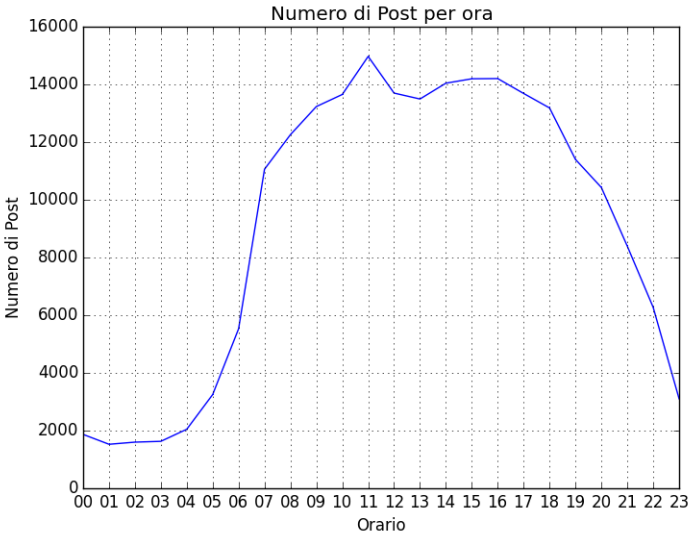

Figura 3.4: Numero di post pubblicati ogni ora

non può essere affermato per quanto riguarda il calo nella produzione osservato dal tardo pomeriggio, mostrato con una forma più morbida nella parte destra del grafico.

Questa differenza osservata può essere causata da alcuni fattori. Sicuramente è da tenere conto che, mentre nelle ore più tarde della notte l'attività sul social è drasticamente più bassa, poiché la maggior parte delle persone dorme, questo può non essere vero nel tardo pomeriggio e nelle prime ore della sera, momenti in cui gran parte delle persone si rilassano ed eventualmente si tengono in contatto con i propri amici attraverso la rete. Inoltre, essendo presi in considerazione in questo caso i post pubblicati da pagine che rappresentano quotidiani nazionali, è chiaro come le prime ore del mattino siano il momento in cui una gran parte dell'informazione viene prodotta e fruita dalle persone. Si tratta infatti dei momenti in cui le redazioni preparano le notizie del giorno.

Anche allo scopo di validare queste ipotesi, è stato prodotto il grafico che rappresenta invece il numero di commenti pubblicati durante la giornata, mostrato in Figura 3.5. Si può chiaramente notare quanto la situazione mostrata sia del tutto simile a quella vista nel caso dei post, con tuttavia alcune differenze degne di nota.

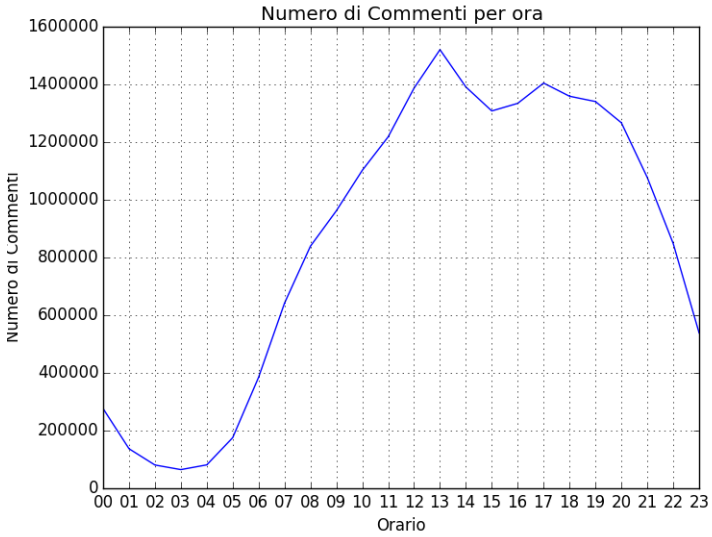

Figura 3.5: Numero di post pubblicati ogni ora

Innanzitutto, Si può notare come la curva sia più morbida da entrambi i lati, senza la presenza di una salita ripida nelle prime ore del mattino come mostrato per i post pubblicati. Inoltre si nota come, generalmente, i picchi e le valli della curva siano in questo caso spostate di circa due ore, sia nella parte centrale della giornata che durante le ore della notte. Ciononostante, come già accennato, la forma della distribuzione all'interno della giornata rimane decisamente molto simile tra post e commenti, come testimoniato da Figura 3.6, dove le due distribuzioni sono state messe a confronto.

Dalla visione comparata delle due distribuzioni è assolutamente evidente come le stesse siano estremamente simili per andamento generale, fatta eccezione per le prime ore del mattino, dove sembra che il numero di post pubblicati salga più rapidamente rispetto al numero di commenti. Va comunque fatto notare come le due distribuzioni siano di ordini di grandezza totalmente differenti, ed è stata infatti usata una scala differente per ognuna delle distribuzioni per poterle visualizzare insieme nel grafico. L'immagine è stata infatti creata semplicemente con lo scopo di mostrare quanto l'andamento giornaliero, sia per la produzione di post che per la produzione dei relativi commenti, fosse estremamente simile.

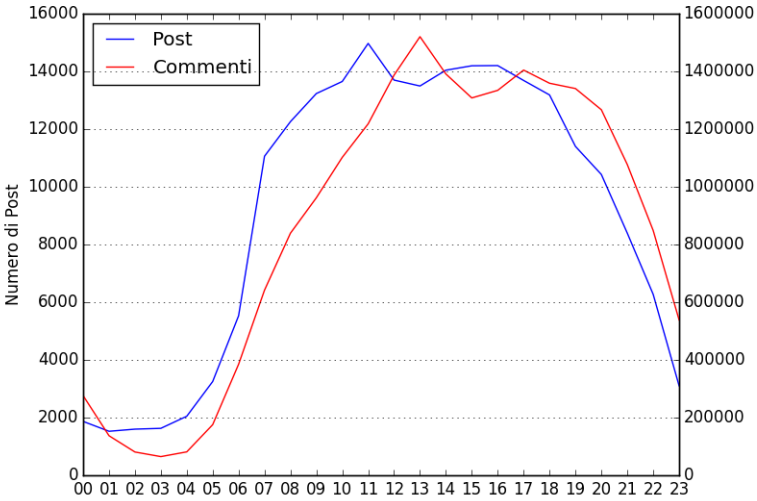

Figura 3.6: Numero di post e numero di commenti pubblicati ogni ora

In conclusione di questa rapida panoramica riguardante alcune dinamiche dei dati a nostra disposizione si può quindi affermare quanto questo tipo di nozioni possano risultare estremamente utili, sia dal punto di vista teorico, per studiare l'andamento dell'attività sul social, sia dal punto di vista pratico, in quanto può essere di aiuto a editori e gestori di pagine Facebook per organizzare al meglio il contenuto da pubblicare durante la giornata.
Inoltre, si può prendere consapevolezza di quanto risorse come quella creata siano al giorno d'oggi sempre più importanti, sia dal punto di vista del Natural Language Processing, come è stato fatto per la parte centrale del presente lavoro, sia per quanto riguarda l'analisi delle reti sociali e il comportamento degli utenti all'interno di esse.

La risorsa è stata denominata FBNEWS-2015, e così come è stata presentata, verrà resa liberamente disponibile.
Nel presente caso con la risorsa è stato creato uno spazio distribuzionale emotivo, che come mostrato nei capitoli successivi, è stato applicato attraverso una serie di feature emotive estratte dal testo, in un classificatore volto ad individuare la polarità, positiva o negativa, di tweet.

Capitolo 4

Esperimenti

Una volta completato il lavoro di creazione del corpus, si è proceduto a effettuare esperimenti volti a constatare se e quanto l'utilizzo di feature emotive potesse risultare di aiuto in un task di classificazione della polarità di documenti.
Per prima cosa quindi, è stato creato uno spazio distribizionale emotivo a partire dal corpus ricavato da Facebook. In seguito, utilizzando come base di partenza lo spazio emotivo, sono state estratte le feature potenzialmente interessanti per l'addestramento del classificatore. Infine, sono stati eseguiti alcuni esperimenti per determinare la combinazione migliore di feature da utilizzare nel classificatore definitivo.
In questo capitolo verranno quindi esaminate le fasi del progetto sopra menzionate.

4.1 Creazione dello spazio distribuzionale emotivo

Per quanto concerne lo spazio emotivo utilizzato, si è deciso di partire dal corpus FBNEWS-2015 e procedere alla creazione dello spazio, attraverso una metodologia di bootstrapping di seed, già testata come efficace specialmente in Passaro et al. (2015)[54] e ispirato da Turney e Littmann (2003)[75].

Il processo completo è descritto in Passaro et. al (2015)[54] per la creazione della risorsa ItEM, a cui si rimanda per una trattazione completa. In questa sede verranno evidenziate le fasi salienti di creazione dello spazio distribuzionale a partire dal

corpus FBNEWS-2015. L'obiettivo è quello di creare un lessico emotivo, sfruttando l'annotazione manuale e spazi distribuzionali. Questo anche perché, in quasi tutte le lingue ad eccezione dell'inglese sono assenti lessici emotivi ad ampia copertura (Passaro et al., 2015)[54].

Per la creazione di ItEM è stata quindi innanzitutto ottenuta, tramite metodologie di crowdsourcing, una serie di parole fortemente legate alle 8 emozioni di Plutchik (1994)[59]. Alle parole ricavate, chiamate *Seed*, sono stati in seguito aggiunti i nomi delle emozioni considerate, così come i loro sinonimi in WordNet, WordNetAffect e nel dizionario Treccani Online. I seed sono il più possibile bilanciati per ogni emozione, e al loro interno per nomi, verbi e aggettivi.

La fase principale del lavoro ha visto poi l'utilizzo di tecniche di Bootstrapping per l'ampliamento della risorsa a partire dai Seed, utilizzando appunto la metodologia distribuzionale proposta da Turney e Littmann (2003)[75]. In particolare, la ricerca si basa sull'assunzione che in uno spazio vettoriale le parole tendano a condividere la stessa connotazione rispetto ai loro vicini. Passaro et al. (2015)[54] hanno dunque estratto da alcuni dei corpora più estesi disponibili per l'italiano, in particolare il corpus di La Repubblica (Baroni et al., 2004) e ItWaC (Baroni et al., 2009) le 30000 parole più frequenti tra nomi, verbi e aggettivi, usati successivamente come target e contesti per una matrice di co-occorrenze estratte all'interno di una finestra di cinque parole centrata sul lemma target. Il calcolo dello score avviene costruendo per ogni coppia $<$ emozione, POS $>$ un vettore centroide dai vettori dei seed appartenenti alla determinata emozione e POS, con un totale quindi di 24 centroidi. Lo score per ogni emozione è stato quindi calcolato in base alla distanza tra i nuovi lemmi e i vettori centroidi, così da ottenenere per ogni parola target un valore per ognuna delle emozioni di base considerate. Per la costruzione del modello a spazi vettoriali è stata utilizzata dapprima la misura di Pointwise Positive Mutual Information (PPMI) come peso degli elementi nella matrice di co-occorrenze. Sono stati poi selezionati i 240 contesti con valori maggiori per ogni parola target, e infine calcolato lo score emotivo per ogni parola target rispetto alle emozioni misurando la similarità del coseno con il

centroide corrispondente. In output viene presentata una lista di parole ordinate in base agli score emotivi (Passaro et al., 2015)[54]. Il processo successivamente è stato validato nuovamente attraverso tecniche di crowdsourcing, in cui è stato chiesto agli utenti di generare uno score emotivo per alcune parole.

Nel caso di FBNEWS-2015, si è quindi deciso di ripetere la fase di bootstrapping presentata, considerando ancora una volta come base di partenza le stesse parole Seed utilizzate per ItEM. In questo caso non è stata effettuata la valutazione tramite crowdsourcing dei valori prodotti, considerando comunque positivo ed efficace l'approccio presentato da Passaro et al. (2015)[54].

Anche in questo caso l'oputput generato è composto da una lista ordinata per emozione e per valore emotivo di parole. In particolare è stata prodotta una lista per ognuna delle tre parti del discorso considerata, cioè nomi, aggettivi e verbi. In tabella è presentato un esempio per i cinque nomi con i valori più elevati per l'emozione gioia.

Emotion	Word	Value
Gioia	gioia-s	0.769305
Gioia	trionfo-s	0.648426
Gioia	soddisfazione-s	0.642233
Gioia	felicità-s	0.636774
Gioia	serenità-s	0.612624

Tabella 4.1: Valori emotivi di gioia più elevati per i nomi

4.2 Dataset utilizzato

Il dataset utilizzato in fase di classificazione è stato ricavato dai dati proposti durante il task di SentiPolC di Evalita 2014. Come spiegato in Basile et al. (2014)[5], il

dataset proposto per la competizione è composto da una serie di tweet provenienti da due corpora, in particolare SENTI-TUT (Bosco et al., 2013)[10] e TWITA (Basile and Nissim, 2013)[6].

Il dataset è composto sia da tweet riguardanti argomenti generici, sia riguardanti in particolare la politica. Questi ultimi sono stati estratti tramite la ricerca di keyword specifiche. Questa informazione è contenuta in un campo riguardante il *topic*.
I dati, nella loro forma base, sono rappresentati all'interno di un file CSV. Un campo è riservato al testo del tweet, mentre negli altri sono rappresentate le informazioni utili al task. In particolare, sono salvati l'ID univoco del tweet, il valore di soggettività, di polarità positiva, polarità negativa, ironia e topic. Al di là dell'ID e del testo, i restanti campi possono assumere valore di 0 o 1, che rispettivamente significano la presenza o l'assenza della feature. Per il campo del topic, 0 rappresenta "generico" e 1 rappresenta "politico" (Basile et al., 2014)[5].

I dati sono stati annotati manualmente prima di essere rilasciati, seguendo alcune specifiche linee guida. Ad esempio, per il task di interesse per la presente ricerca, un tweet annotato come "oggettivo" non avrà mai valori per la polarità, mentre un tweet soggettivo può esprimere, oltre a una polarità spiccatamente positiva o negativa, sia una polarità mista, ovvero valore 1 sia per polarità positiva che negativa, che nessuna polarità, ovvero 0 per entrambe le feature di polarità. Infine, un tweet marcato come "ironico" deve sempre essere soggettivo ed esibire una specifica polarità.

I tweet rilasciati come training-set da Evalita sono 4513. Tuttavia, date alcune restrizioni imposte da Twitter, enti esterni possono fornire solamente gli ID dei suddetti tweet, che possono quindi essere rimossi per varie ragioni (volontà dell'utente, sospensione dell'account e simili) in ogni momento. Nel caso dei dati utilizzati, non sono risultati disponibili 493 tweet.
Di seguito è riportato un esempio del file csv contenente i tweet e i valori necessari per la valutazione della classificazione.

	A	B	C	D	E	F	
1	idtwitter	subj	pos	neg	iro	top	TEXT
2	193201249095651000	1	0	1	0	1	#Grillo Mi fa paura la gente che urla. Ne abbiamo già visti almeno un paio, ed è finita com'è finita. Niente urla per me, grazie.
3	139741158778749000	1	0	1	0	1	Domattina Forza Nuova farà un banchetto in piazza a Faenza per dimostrare contro il governo Monti: "No a governo di banchieri".
4	193641338066059000	0	0	0	0	1	Anche allo stadio si canta #Grillo uno di noi
5	143731095144370000	0	0	0	0	1	(LEO) Come Volevasi Dimostrare: Con il cattolicissimo Governo golpista di Mario Monti ed il tacito consenso di un... http://fb.me/1jt49IkoJ
6	187993229222883000	1	1	0	0	0	@Ale__Malik oddio quanto ti capisco.<3 *--*

Figura 4.1: Esempio di Tweet e valori per la classificazione del training-set Evalita

I dati tuttavia, nella forma presentata da Evalita, risultanto essere difficilmente utilizzabili in un'ottica di estrazione di feature, soprattutto nel caso in cui queste siano fortemente legate a valori per parole singole, come si vedrà nella successiva sezione. Proprio per questo è stata necessaria una fase di pre processing, volta principalmente ad effettuare la tokenizzazione, il sentence splitting e il POS-tagging dei tweet a disposizione.
Come anche per i dati riguardanti il corpus estratto da Facebook, analizzato nel Capitolo 3, sono state quindi eseguite queste procedure di pre processing, oltre a codificare le informazioni presenti in maniera differente, per rendere più semplice l'estrazione. Di seguito è riportato l'esempio di un tweet dopo il preprocessing.

```
<doc id="160103675598077953" polarity="01">

1  Cosa  cosa  P  PQ num=s|gen=n
2  intendo intendere  V  V  num=s|per=1|mod=i|ten=p
3  fare  fare  V  V  mod=f
4  per   per   E  E  _
5  i  il R  RD num=p|gen=m
6  poveri  povero  S  S  num=p|gen=m
7  ed e  C  CC _
8  i  il R  RD num=p|gen=m
9  bisognosi  bisognoso  S  S  num=p|gen=m
10 ?  ?  F  FS _

1  Una   uno   R  RI num=s|gen=f
```

```
2  beata beato S S num=s|gen=f
3  minchia minchia S  S  num=n|gen=n
4  .  .  F  FS _

1  Cetto Cetto S SP _
2  La il R RD num=s|gen=f
3  Qualunque  qualunque  D  DI num=s|gen=n
4  ?  ?  F  FS _

1  No no B BN _
2  Mario Mario S SP _
3  Monti Monti S SP _

</doc>
```

Come si può notare, in questa fase sono mantenute solo le informazioni riguardanti l'ID del tweet e la sua polarità, oltre al testo. Inoltre, la polarità è salvata come la concatenazione dei due campi presenti nel documento originale, per conformarla allo standard per la valutazione. In questo caso ad esempio, lo 01 rappresenta un valore di polarità positiva pari a 0, e un valore negativo pari a 1, a significare la negatività del tweet. Risulta importante precisare che le restanti informazioni, pur non essendo fondamentali per il task di sentiment polarity classification, sono comunque state mantenute e utilizzate in seguito perché richieste dagli strumenti di valutazione, sia per la fase di training che di test.
In seguito al preprocessing quindi, si può osservare come i token siano divisi per frase e a ognuno di essi sia associata la parte del discorso corrispondente, in varie forme.

4.3 Estrazione delle feature emotive

In seguito alla creazione dello spazio emotivo a partire dai dati ricavati da Facebook, ci si è concentrati sull'estrazione delle feature emotive da aggiungere a quelle utilizzate per il task di SentiPolC di Evalita dal Laboratorio di Linguistica Computazionale dell'Università di Pisa (Passaro et al, 2014)[53].
Come già evidenziato, ci si è concentrati sui valori emotivi delle parole riportate all'interno dei tweet proposti da Evalita.

4.3.1 Feature emotive generali

In primo luogo si è cercato di individuare alcune feature generali, che potessero rappresentare ed esprimere il livello di emotività del Tweet analizzato.
In particolare, sono state estratte le seguenti feature:

Numero di parole emotive Numero di parole contenute nel tweet presenti all'interno dello spazio emotivo di Facebook.

Rapporto parole emotive/totale Numero di parole emotive sul totale delle parole presenti nel tweet.

Numero di parole fortemente emotive Numero di parole emotive, opportunamente filtrate in modo da considerare solamente quelle più emotive.a questo scopo è stata impostata una soglia minima di 0.4 per almeno uno degli otto valori emotivi di ogni parola.

Score di emotività Score volto a rappresentare il livello di emotività generale di ogni tweet. È calcolato come il rapporto tra il numero di parole fortemente emotive e il numero di parole contenuto. Assume valori che possono variare all'interno dell'intervallo [0, 1]. In caso di assenza di parole fortemente emotive o parole contenuto lo score è stato impostato automaticamente a 0.

$$Emotivity(Tweet) = \frac{Count(Parole_fortemente_emotive)}{Count(parole_contenuto)} \tag{4.1}$$

4.3.2 Valori massimi per le emozioni

Per ogni emozione è stato calcolato il valore massimo assunto all'interno del tweet. In particolare, ognuna delle 8 feature estratte rappresenta una delle emozioni prese in considerazione, ed è conservato per ciascuna il valore massimo tra quelli riscontrati tra i valori di ogni parola emotiva presente nel tweet.

4.3.3 Quartili

Attraverso questa serie di feature si è cercato di tenere in considerazione la distribuzione di tutte le parole emotive rispetto a ogni singola emozione.
Innanzitutto, per ogni emozione, la lista delle parole emotive è stata ordinata per valori decrescenti e suddivisa in quartili. In questo modo le parole emotive presenti nello spazio sono ordinate e suddivise per ogni emozione, cosicché il quarto quartile rappresenti la lista delle parole con i valori più alti e il primo quartile la lista delle parole con i valori meno elevati.
Successivamente sono state generate 32 feature, ovvero 4 quartili per ognuna delle 8 emozioni. Per ogni parola emotiva riscontrata all'interno del tweet, è stato incrementato il corrispondente valore del conteggio per il quartile dove è presente per ognuna delle emozioni.

4.3.4 Seed di ITEM

La risorsa ITEM è stata utilizzata come base per la creazione dello spazio emotivo derivato dai dati di Facebook. Proprio per questa ragione è parso interessante includere tra le feature emotive anche la parte fondamentale di questa risorsa, ovvero le parole *seed*. Si tratta di una serie di parole, in particolare 347, estratte da Passaro et al. (2015)[54] attraverso un paradigma di crowdsourcing online, e corredate con i valori emotivi per ogni emozione, generati dal calcolo della frequenza del termine rispetto al numero di emozioni per cui è stato generato e utilizzati come base per effettuare il bootstrapping dello spazio emotivo attraverso del modello presente in Turney e Littman (2003)[75].

Si è deciso, per quanto concerne le feature, di considerare 4 parole seed per ogni emozione, in particolare quelle con la frequenza calcolata più alta all'interno del set di tweet fornito come addestramento. Sono state quindi generate anche in questo caso 32 feature, 4 per ognuna delle 8 emozioni, per le quali i valori rispecchiano la presenza o meno della parola seed all'interno del tweet.

4.3.5 Parole distintive per ogni emozione

Come per le parole seed frequenti, si è deciso di procedere considerando anche le parole ritenute più distintive per ognuna delle emozioni.
Per calcolare il valore di *distintività* di una parola rispetto ad una emozione, si è proceduto a creare una differente versione dello spazio distribuzionale ricavato dal corpus di Facebook, per il quale i valori associati a ognuna delle parole per le emozioni sono stati normalizzati. In particolare, per ogni parola target sono stati presi in considerazione i valori di tutte le emozioni. Questi sono stati successivamente sommati. Il valore così ottenuto è stato utilizzato per ricalcolare, con una proporzione del tipo $ValoreBase : SommaTotale = ValoreNormalizzato : 1$, il valore assegnato a ogni emozione, così da ottenere una sorta di valore percentuale per ognuna di esse. In questo modo, la somma totale dei nuovi valori ammonta a 1, e ogni valore rappresenta in percentuale la presenza di una emozione rispetto alle altre per il termine. Un esempio è mostrato nel diagramma a torta per la parola INSENSIBILE a seguito della normalizzazione.

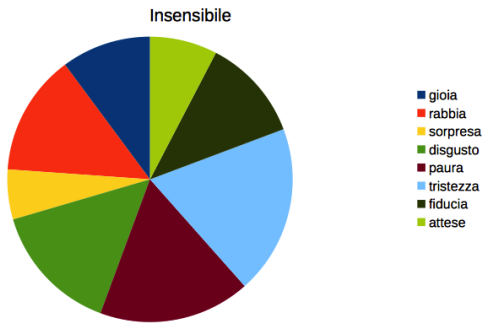

Figura 4.2: Valori emotivi normalizzati per la parola *insensibile*

La normalizzazione è necessaria in quanto nello spazio iniziale, le parole rappresentante non hanno valori comparabili per le varie emozioni. Questo significa che la somma totale delle emozioni non è equivalente a 1, e non è certo che per ogni emozione il valore massimo sia 1. Questo è causato all'interno dello spazio distribuzionale dallo spread dei coseni per le varie emozioni. In particolare, il valore massimo per due emozioni differenti può variare molto, a seconda della vicinanza delle parole ad un centroide rispetto all'altro. Ad esempio, le parole più vicine al centroide FIDUCIA possono comunque essere più lontane rispetto alle parole più vicine al centroide RABBIA, poiché i valori emotivi non sono assoluti ma relativi al centroide di riferimento. In seguito alla normalizzazione dunque i nuovi valori dovrebbero essere meno accurati dal punto di vista strettamente spaziale, ma più veritieri nel considerare una parola distintiva o meno per una particolare emozione. Ad esempio, se una parola ha un valore emotivo per RABBIA molto elevato in percentuale rispetto alle altre emozioni, questa può con maggiore sicurezza essere considerata come una parola particolarmente distintiva per una determinata emozione. Al contrario, se ha valori simili per più emozioni differenti, come nell'esempio mostrato in figura 4.1, la parola ha meno possibilità di essere rappresentata come distintiva di una delle due emozioni. Per calcolare le parole distintive da scegliere per ogni emozione si è dunque considerato il gruppo di parole con i più elevati valori emotivi nello spazio normalizzato. La normalizzazione assicura che i valori più elevati, proprio perché rappresentati come una percentuale sul totale per la parola, siano con molto bassa probabilità rivaleggiati da valori altrettanto forti. Per fare un esempio, se una parola ha valore emotivo normalizzato per PAURA di 0.8 (dunque 80% del totale), le altre 7 emozioni avranno necessariamente valori più bassi, e la parola sarà tra le più distintive per l'emozione PAURA. Al contrario, se un termine ha valori simili sebbene elevati per due emozioni, ad esempio 0.4 e .039, questo non verrà considerato come distintivo per nessuna delle due emozioni. Avendo a disposizione 3 spazi diversi, ognuno rappresentante una determinata parte del discorso (nomi, verbi o aggettivi), si è deciso di calcolare i valori soltanto per la loro unione.

Sono state quindi prodotte 32 feature: 4 parole distintive per ognuna delle 8 emozioni.

4.3.6 Conteggio di parole positive e negative

Successivamente si è deciso di produrre due feature volte a rappresentare il numero di parole positive e negative riscontrate all'interno del tweet.

Al fine di individuare la polarità di una parola si è deciso di utilizzare una formula che tenesse conto dei valori emotivi. Per questo, si è deciso di considerare alcune emozioni come aventi una polarità strettamente positiva, e altre come aventi polarità strettamente negativa. Le emozioni considerate positive sono *gioia* e *fiducia*, mentre al contrario *disgusto*, *rabbia*, *paura* e *tristezza* sono state considerate negative. Le due emozioni rimanenti nello spazio, *attese* e *sorpresa*, proprio per il loro carattere evidentemente ambiguo, non sono state considerate. Questo può essere anche osservato sperimentalmente. Per testare l'ipotesi infatti è stata utilizzata la risorsa Sentix (Basile e Nissim, 2013)[6], un lessico di parole italiane annotato con valori di polarità negativa e positiva. Per ognuna delle parole presenti nello spazio emotivo creato tramite il corpus di Facebook, è stata controllata la presenza in Sentix. In caso affermativo, si è proceduto ad individuare quali fossero le emozioni maggiormente correlate alla parola. Per fare ciò si è controllato, per ogni emozione, se la parola facesse parte del terzo o del quarto quartile, ovvero i quartili contenenti le parole con valori maggiormente elevati. In caso affermativo, si è valutato quale dei valori presenti in Sentix per la polarità fosse il maggiore. Infine, la parola è stata aggiunta al conteggio delle parole positive o negative per quell'emozione, a seconda di quale fosse il valore più elevato.

Una volta eseguito il processo è stato effettuato il conteggio delle parole positive e negative rilevate per ciascuna emozione per definire la polarità della stessa. Attraverso l'utilizzo di questo metodo è risultato chiaro come le emozioni chiaramente negative e chiaramente positive siano state individuate correttamente, mentre le due emozioni più ambigue, attese e sorpresa, presentano uno scarto molto ridotto tra il numero di parole positive e negative rilevate, a riprova della correttezza dell'ipotesi volta a escluderle dal calcolo della polarità.

La formula utilizzata per il calcolo del valore di polarità totale della parola è la seguente:

$$Polarity(Word) = \frac{Gioia + Fiducia}{2} - \frac{Disgusto + Paura + Rabbia + Tristezza}{4} \tag{4.2}$$

Per ogni parola è stato quindi calcolato il valore di polarità, che può risultare positivo, ovvero maggiore di 0, o negativo, quindi minore di 0. Infine, viene effettuato il semplice conteggio delle parole con polarità negativa e positiva incontrate all'interno di un tweet. I valori così ottenuti sono rappresentati dalle due feature.

4.3.7 Emozioni più prominenti nel tweet

Con questo set di feature si è cercato di individuare il tipo di emotività espresso dalle parole con valori emotivi più alti presenti all'interno dei tweet.

In particolare, anche in questo caso è stato utilizzato lo spazio distribuzionale normalizzato, precedentemente illustrato a scopo di individuare le parole distintive. L'idea è stata quella di ordinare, all'interno del tweet, le parole emotive presenti in ordine decrescente per emotività, cioè dalla più distintiva alla meno distintiva. In seguito, per ognuno dei termini individuati, è stato individuato il valore di distintività massimo, ovvero l'emozione prominente per la parola in questione, per le prime 4 parole individuate in questo modo. Per ragioni di compatibilità e coerenza sia con le altre feature estratte che con il classificatore, si è scelto tuttavia di non utilizzare feature nominali (ovvero la stringa contenente il nome dell'emozione). Al contrario, si è dunque deciso di creare una feature binaria per ogni emozione, per ognuna delle prime quattro parole più emotive, che rappresenta per ogni parola quale delle otto emozioni è quella maggiormente distintiva. Le feature prodotte in totale sono quindi 32.

4.3.8 Polarità delle parole emotive

Infine, con l'ultimo set di feature si è voluta rappresentare l'informazione riguardo la polarità delle parole emotive riscontrate all'interno del tweet. Questo può risultare di rilevanza, poiché lo scopo della classificazione è effettivamente quello di individuare la polarità più che l'emotività di un tweet. L'utilizzo di informazione di tipo emotivo tuttavia può essere molto utile a tale scopo.

Si è quindi deciso di produrre un numero di feature pari al massimo di parole emotive riscontrate nel test set di tweet a disposizione. Le feature sono quindi 20.

Ogni feature rappresenta la polarità delle parole emotive incontrate all'interno del tweet, in ordine di apparizione. La polarità è calcolata come per il conteggio di parole positive e negative. Le parole negative avranno un punteggio inferiore a 0, le parole positive un punteggio superiore.

Va fatto notare come le feature rappresentanti la polarità siano state aggiunte in un secondo momento. Per questo motivo non sono presenti in tutti gli esperimenti riguardanti il training set. Verrà fatta una chiara distinzione tra gli esperimenti utilizzanti la polarità e gli esperimenti nei quali non è stata considerata. Per quanto riguarda invece i risultati presentati a proposito del test set, essa è sempre stata utilizzata.

4.3.9 Estrazione

Le feature sono state estratte tramite uno script in Python. L'algoritmo dapprima alloca una serie di oggetti e definisce una serie di funzioni volte a riconoscere gli elementi emotivi all'interno del tweet. In seguito l'intero test-set, opportunamente etichettato con le Part-Of-Speech corrispondenti, viene analizzato, e le feature emotive di ogni tweet sono salvate all'interno di un file CSV per la successiva classificazione.
Il codice utilizzato è disponibile in appendice.

L'algoritmo, per quanto funzionale, è risultato nella pratica non troppo rapido. Risulterà quindi un punto fondamentale per eventuali analisi successive il potenziamento delle prestazioni dell'algoritmo.

L'output è dunque un file CSV contenente l'ID univoco di ogni tweet rilasciato da Evalita per il test set, e una serie di 163 colonne, corrispondenti ad altrettante feature estratte dai tweet.

4.4 Creazione dei modelli

In questa sezione verrà mostrato il workflow tecnico utilizzato per effettuare gli esperimenti di classificazione sul dataset di tweet. In particolare, verranno presentati il tool usato per la classificazione e gli algoritmi implementati. Considerando come base di partenza e come termine di paragone il lavoro svolto da Passaro et al. (2014)[53], si è deciso di seguire esattamente il workflow presentato nello studio.

Per la classificazione si è deciso di utilizzare il tool di data analysis Weka (Hall et al., 2009)[24]. SI tratta di una collezione di algoritmi di machine learning applicabili a vari task di data mining. Gli algoritmi possono essere sia utilizzati direttamente tramite il tool e la sua interfaccia visuale, sia implementati attraverso codice Java. Si è deciso nel presente caso, oltre che per semplicità, di utilizzare l'interfaccia grafica, poiché permette di interagire direttamente coi dati e scegliere velocemente il subset di feature da utilizzare per gli esperimenti di classificazione e modificarlo facilmente senza la necessità di interagire direttamente con i file di input originali. Oltre a strumenti di classificazione, Weka è fornito anche di tool per il pre-processing dei dati, regressione, clustering, regole associative e visualizzazione (Hall et al., 2009)[24].

L'interfaccia visiva molto semplice e intuitiva, come mostrato in figura, permette appunto la modifica del set di feature da utilizzare, attraverso la gestione di file di tipo ARFF presi in input. Attraverso la finestra dedicata alla classificazione, con una serie di sotto-menù, è possibile scegliere gli algoritmi di classificazione da

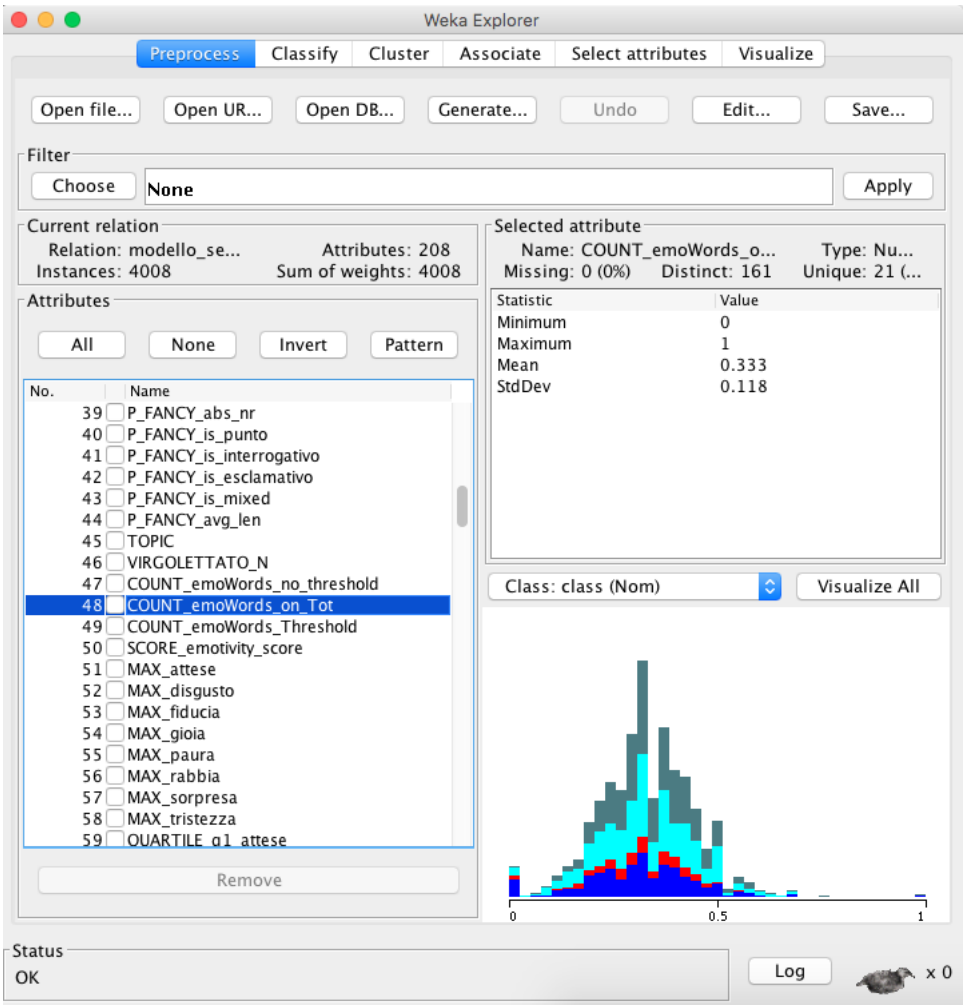

Figura 4.3: Finestra di pre-processing di Weka

utilizzare, modificarne le impostazioni, impostare le preferenze di output, oltre ad avere una finestra dedicata appunto alla classificazione, con informazioni dettagliate sui processi svolti dall'algoritmo (es. pesatura delle feature) e i risultati ottenuti.

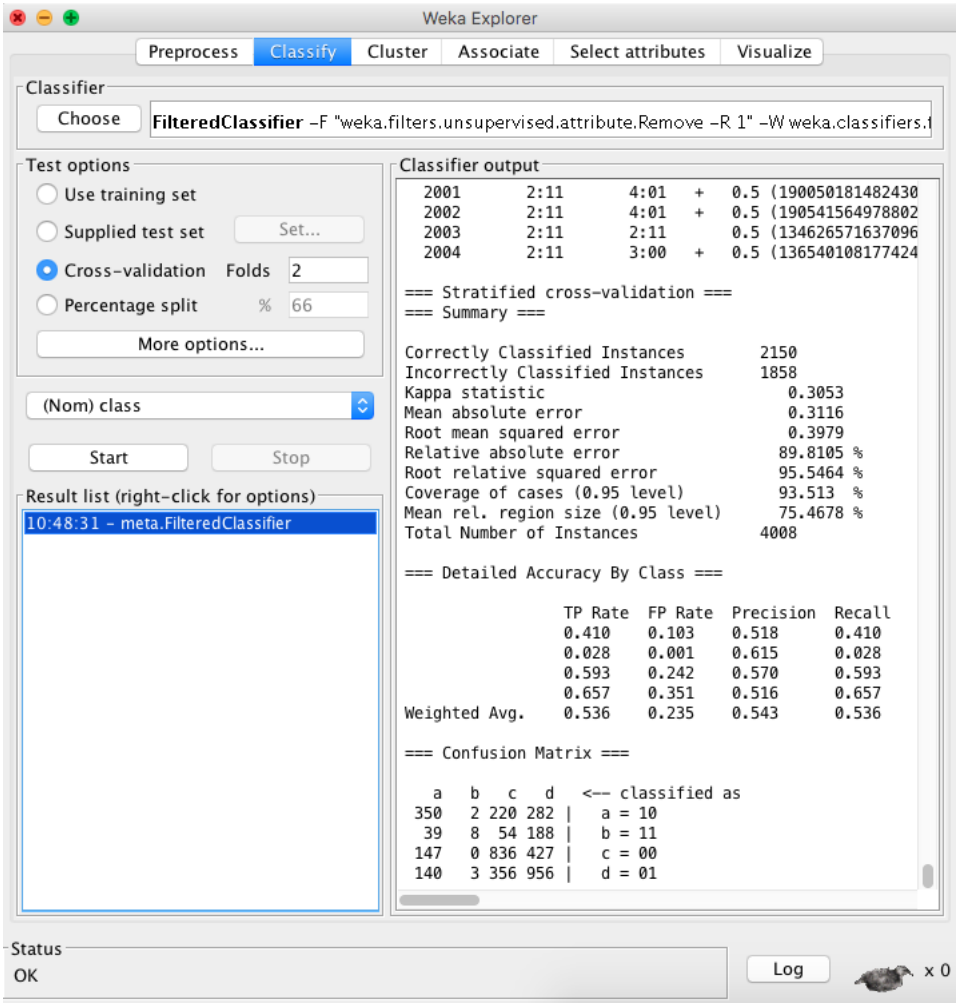

Figura 4.4: Finestra di classificazione di Weka

Per quanto riguarda l'algoritmo utilizzato, ancora una volta come in Passaro et al. (2014)[53], si è scelto un classificatore SVM con una implementazione lineare del kernel disponibile in Weka (Witten et al., 2011)[84], addestrato attraverso l'algoritmo di ottimizzazione sequenziale minima, *Sequential Minimal Optimization(SMO)* introdotto da Platt (1998)[58]. Di seguito è riportato il codice Java creato attraverso Weka per lanciare l'algoritmo di classificazione.

```
weka.classifiers.meta.FilteredClassifier -F
    "weka.filters.unsupervised.attribute.Remove -R 1" -W
    weka.classifiers.functions.SMO -- -C 1.0 -L 0.001 -P
    1.0E-12 -N 0 -V -1 -W 1 -K
```

```
"weka.classifiers.functions.supportVector.PolyKernel -C
250007 -E 1.0"
```

4.5 Esperimenti effettuati

Dopo aver evidenziato i metodi attraverso i quali è stata tecnicamente effettuata la fase di classificazione dei tweet, risulta necessario mostrare, nella pratica, quali esperimenti sono stati effettuati.

Si è deciso, dopo i primi esperimenti, di procedere secondo una logica volta a evidenziare eventuali differenze nelle prestazioni del classificatore riscontrate a seconda dello spazio distribuzionale utilizzato. In particolare, si è deciso di creare tre feature-set differenti sulle quali addestrare l'algoritmo. Le feature estratte sono le medesime in ognuno dei casi, mentre sono stati modificati di volta in volta i valori presenti all'interno dello spazio distribuzionale. Questi non è stato effettivamente ricalcolato attraverso un processo differente, ma sono stati normalizzati i valori già presenti appartenenti ad esso in due modi differenti. In particolare, oltre agli esperimenti con lo spazio di base, sono stati effettuati anche esperimenti utilizzando uno spazio normalizzato attraverso la misura di *z-score* e attraverso una normalizzazione dei valori emotivi per ogni parola. Saranno quindi presentati nel dettaglio gli esperimenti effettuati con le tre differenti variazioni sui valori dello spazio distribuzionale, oltre che, per ognuno di essi, esperimenti con feature set leggermente differenti, volti a comprendere quale fosse la combinazione di feature più efficace per la classificazione.

Gli esperimenti presentati in questa sezione sono inoltre stati svolti solamente considerando i valori ottenuti utilizzando soltanto il training set come corpus, e la tecnica di *k-fold cross validation* per la valutazione delle prestazioni. Questo approccio consiste nell'esecuzione progressiva di una serie di run dell'algoritmo di classificazione, ognuna delle quali riserva una percentuale differente del corpus da utilizzare come test set. In particolare, così facendo ogni record presente nel dataset è utilizzato lo

stesso numero di volte per l'addestramento ed esattamente una volta per il test (Tan et al., 2005)[72].
Il dataset viene quindi diviso in k partizioni di uguale dimensione. Durante ogni esecuzione, una partizione k viene utilizzata per il test mentre le altre sono utilizzate per l'addestramento. L'algoritmo esegue questa procedura k volte, in modo da utilizzare ogni segmento k esattamente una sola volta per il test. L'errore totale è in seguito calcolato attraverso la somma totale degli errori di ognuna delle run (Tan et al., 2005)[72].
Combinando in questo modo i risultati, è possibile addestrare il classificatore sull'intero corpus di addestramento, ma anche di ottenere una classificazione sullo stesso dataset nella sua interezza, anziché utilizzarne solamente una piccola parte. Si è quindi in grado di valutare con maggiore sicurezza le prestazioni del classificatore, poiché in assenza di un test set si sarebbe altrimenti costretti ad addestrare il classificatore solamente su parte del dataset, ed eseguire la classificazione sulla parte restante. Pur avendo a disposizione anche il test set, si è voluto ricreare le condizioni dell'esperimento originale, e in generale della maggior parte di esperimenti di questo tipo: si valutano inizialmente le performance del classificatore sui dati a disposizione per scegliere le migliori combinazioni di feature possibili, per poi valutare i modelli ottenuti tramite la classificazione del test-set.

Nel presente caso, la cross-validation è stata eseguita attraverso l'utilizzo di 10 *fold*. La classificazione è stata quindi effettuata attraverso 10 run successive, ognuna volta ad addestrare il classificatore sul 90% dei dati a disposizione e testarla sul rimanente 10%. Weka permette attraverso un menu di scegliere direttamente questo approccio per la classificazione, e aggrega automaticamente i dati risultanti fornendo a schermo il risultato complessivo della classificazione.

Il task proposto da Evalita 2014 ha messo a disposizione la possibilità di sfruttare sia un classificatore multiclasse, cioè un classificatore in grado di riconoscere le quattro polarità, *positiva*, *negativa*, *nessuna polarità* e *polarità mista*, che la combinazione di due classificatori binari addestrati su due modelli differenti, uno per la classe posi-

tiva e uno per la negativa, da unire in fase di valutazione. Nell'esperimento proposto da Passaro et al. (2014)[53] entrambi gli approcci sono stati valutati e testati in fase di sviluppo, ma non hanno mostrato differenze significative nei risultati. Per questo, sia per Passaro et al. (2104)[53] che per la presente ricerca, si è deciso per l'utilizzo del classificatore multiclasse, in quanto più rapido e semplice.

Risulta necessario inoltre presentare il criterio attraverso il quale sono stati valutati i modelli presentati. Chiaramente, trattandosi di un problema di classificazione, sono state utilizzate le misure classiche, ovvero *Precision*, *Recall*, e *F-measure*. Nel caso particolare della classificazione sono utili per valutare la bontà di un modello attraverso la valutazione dei casi in cui questi risulta corretto o sbagliato.

Precison La Precision è calcolata come segue:

$$Precision = \frac{tp}{tp + fp}$$

Questa misura può essere considerata come una misura di esattezza del classificatore. Misura infatti il numero di istanze positive classificate correttamente come tali rispetto al numero totale di istanze individuate come positive. La misura fornisce appunto una quantificazione della precisione del classificatore nell'individuazione delle istanze. Ad esempio, nel caso particolare dei tweet, potrebbe essere il rapporto tra il numero di tweet positivi individuati correttamente e il numero di tweet positivi individuati in totale.

Recall Per il calcolo della Recall, la formula utilizzata è:

$$Recall = \frac{tp}{tp + fn}$$

La Recall, a differenza della Precision, misura invece il numero di istanze classificate correttamente come positive rispetto al numero di istanze positive presenti nel dataset. Ad esempio, il numero di tweet individuati correttamente in quanto positivi sul numero totale di tweet positivi presenti nei dati a disposizione.

F-measure La F-measure, o F-score bilanciato, infine si calcola dunque come:

$$F = 2 \cdot \frac{Precison \cdot Recall}{Precision + Recall}$$

Questa è la media armonica tra le due misure di Precision e Recall, e risulta utile in caso di voler valutare con una singola misura la prestazione del sistema di classificazione preso in considerazione. Proprio per questo è stata utilizzata ad esempio come misura univoca di scoring per i classificatori presentati dai candidati nell'ambito del task di Sentiment Polarity Classification di Evalita 2014 (Basile et al., 2014)[5].

Tuttavia, va notato come si è cercato nel caso della presente ricerca di incrementare in modo particolare le prestazioni riguardanti la recall riscontrata nel classificatore positivo, che come mostrato nel capitolo 2.4.1, si è rivelato essere il principale punto di debolezza per il classificatore di partenza. Questo, come spiegato in Passaro et al. (2014)[53], oltre al fatto che la distribuzione di classi nei dataset di tweet per addestramento e test è sbilanciata, può essere causato dal fatto che le feature lessicali positive sono probabilmente non così efficaci nella predizione della positività del tweet rispetto alle feature negative per tweet negativi. Pur non considerando come baseline per la presente ricerca il modello in cui sono presenti anche le feature lessicali, anche nel modello cosiddetto Shallow le prestazioni per quanto riguarda il classificatore positivo su tweet positivi risultano essere meno accurate rispetto agli altri valori.
Una prima scrematura dei risultati ottenuti, così come mostrato nelle prossime sezioni, è stata quindi effettuata in buona parte sulla base delle prestazioni di recall su tweet positive, ovvero sono stati scelti inizialmente i modelli con valori più alti per questa caratteristica.

4.5.1 Spazio emotivo standard

I primi esperimenti sono stati condotti sulla versione standard dello spazio emotivo, ovvero con i valori emotivi per ogni parola estratti così come spiegato nel capitolo 4.3.

Si è deciso di testare, trattandosi dei primi esperimenti svolti in ordine temporale, varie configurazioni, volte a evidenziare possibili punti di forza o debolezza delle feature. Risulta infatti necessario comprendere quali delle feature aggiunte al modello considerato come baseline possono essere di aiuto nella classificazione e quali possano essere escluse in quanto capaci di generare prestazioni inferiori o non in grado di fornire un valore aggiunto per la classificazione.

Sono state quindi testate varie configurazioni, rimuovendo e aggiungendo set di feature con caratteristiche simili tra loro. Oltre a testare le feature prodotte specificamente per questa ricerca, cioè le feature ricavate dallo spazio emotivo, sono stati svolti esperimenti volti anche a individuare eventuali conflitti tra queste e le feature presenti nel modello di base, senza ottenere risultati positivi. Il modello di base prodotto da Passaro et al. (2014)[53] è rimasto quindi inalterato. Le varie configurazioni proposte con i rispettivi risultati, sono presenti in appendice. In questa sede ci si limiterà a mostrare i risultati ottenuti in fase di sviluppo e addestramento per i modelli migliori.
I modelli migliori in questo caso sono risultati essere quattro.

All Features Il primo modello presentato è quello contenente tutte le feature estratte, oltre anche alle feature presenti nel modello baseline.
Le feature utilizzate sono quindi:

- Feature del modello Baseline
- Conteggio parole positive e negative
- Score di emotività
- Massimi per ogni emozione
- Quartili
- Parole Seed di Item per ogni emozione (4)
- Parole distintive per ogni emozione (4)

- Conteggio di parole positive e negative
- Parole più emotive per ogni emozione (4)

I risultati ottenuti sul training set da questo modello sono presentati in tabella. Dei modelli presentati facenti utilizzo dello spazio emotivo di base, è risultato essere il migliore in assoluto.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.73	0.68	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.2: Risultati del training-set sul modello All Features

No Score Per la creazione di questo modello, come si evince dal nome, non è stata utilizzata la feature dello score di emotività. I risultati sono molto simili a quelli presentati per il modello con tutte le feature.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.36	0.45
NEG	0	0.73	0.68	0.71
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.3: Risultati del training-set sul modello No Score

No Distinctive In questo caso si è deciso di non considerare l'intero set di feature rappresentanti le parole più distintive per ogni emozione. Anche in questo caso, le

differenze con il modello avente tutte le feature non sono in alcun modo significative. Tuttavia, è interessante notare come la recall per il task POS, classe 1, la più interessante per questa ricerca, subisca un decremento anche se molto leggero, di un solo punto percentuale. Al netto dell'assenza di 32 feature (4 parole distintive per ognuna delle 8 emozioni) sembra comunque un risultato interessante.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.35	0.45
NEG	0	0.74	0.69	0.71
	1	0.63	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.4: Risultati del training-set sul modello No Distinctive

Come già evidenziato, per quanto riguarda i modelli prodotti senza alcun tipo di normalizzazione per quanto concerne i valori presenti nello spazio distribuzionale, il migliore in assoluto è risultato essere comunque quello in cui tutte le feature sono state utilizzate.

Nel capitolo 4.5.4 saranno mostrati i risultati degli stessi modelli aggiungendo le feature della polarità. Tuttavia, prima risulta necessario mostrare anche gli altri modelli prodotti senza l'utilizzo della polarità, creati normalizzando lo spazio emotivo.

4.5.2 Normalizzazione per parola

Nel secondo gruppo di esperimenti si è deciso dunque di ripetere alcuni degli esperimenti con i migliori risultati del primo gruppo, tuttavia modificando i valori dello spazio emotivo. In particolare, si è deciso di normalizzare i valori presenti nello spazio distribuzionale andando a considerare la distribuzione dei valori per ogni singola parola, esattamente come eseguito nel caso delle feature riguardanti le parole

distintive per ogni emozione. Se tuttavia nel caso precedente la normalizzazione è stata effettuata semplicemente per il calcolo della distintività delle feature, nel caso degli esperimenti è stato utilizzato lo stesso spazio distribuzionale normalizzato sia per il calcolo della distintività che per recuperare i valori emotivi di ogni parola incontrata all'interno dei tweet. Si riporta un esempio del codice utilizzato per effettuare la normalizzazione dello spazio, in particolare la parte necessaria a calcolare i nuovi valori da inserire rispetto alle emozioni di ognuna delle parole presenti.

```
for line in emo_space_S:
   line.strip()
   splitted = line.split("\t")
   emotion = splitted[0]
   word = splitted[1]
   value = splitted[2].strip()

   emolex_S[word][emotion] = unicode(value)

unordered_list = []
for word, valori in emolex_S.iteritems():
   somma = 0
   for emotion, value in emolex_S[word].iteritems():
   somma+=float(value)

   for emotion, value in emolex_S[word].iteritems():
      new_value = float(value) / somma
      emolex_S[word][emotion] = new_value
```

Va notato come nel caso particolare presentato sia utilizzato lo spazio emotivo riguardante i nomi. La stessa operazione è stata chiaramente effettuata anche per gli aggettivi e i verbi.

Anche in questo caso, i migliori tre esperimenti sono risultati essere quelli condotti utilizzando tutte le feature, rimuovendo lo score di emotività e rimuovendo le feature distintive

All Features L'esperimento in cui sono state considerate tutte le feature prodotte anche in questo caso è risultato essere il migliore, come si può osservare in tabella. In particolare, si nota come rispetto all'esperimento All Features standard, in questo caso soprattutto i valori per i task POS:1 e NEG:1 siano risultati leggermente migliori.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.37	0.46
NEG	0	0.74	0.69	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.5: Risultati del training-set sul modello normalizzato per parola All Features

No Score Il secondo modello presentato anche in questo gruppo è stato creato nuovamente eliminando la feature riguardante lo score della distintività. Anche in questo caso, si è notato un leggerissimo aumento dei valori per quanto concerne l'individuazione della classe 1 in entrambi i task rispetto al caso base.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.36	0.45
NEG	0	0.74	0.68	0.71
	1	0.63	0.69	0.66
GLOBAL		0.71	0.72	0.71

Tabella 4.6: Risultati del training-set sul modello normalizzato per parola No Score

No Distinctive Infine sono mostrati anche i risultati riguardanti l'esperimento condotto sul modello in assenza delle feature rappresentanti le parole distintive per ogni emozione. Si nota ancora una volta un leggero aumento per quanto riguarda i

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.37	0.46
NEG	0	0.74	0.68	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.7: Risultati del training-set sul modello normalizzato per parola No Distinctive

valori di Recall per le classi 1 di entrambi i task, mentre gli altri valori rimangono generalmente molto simili all'esperimento originale.

Per la lista completa, nonché i risultati, di tutti gli esperimenti effettuati utilizzando lo spazio emotivo normalizzato per parola si rimanda all'appendice.

4.5.3 Normalizzazione per z-score

Il secondo tipo di normalizzazione effettuata ha riguardato ancora una volta lo spazio emotivo. In questo caso, si è deciso di modificare i valori utilizzando la misura di normalizzazione di *z-score*.

Lo z-score, altrimenti detto standard score, indica all'interno di una distribuzione quante deviazioni standard sussistono tra l'elemento e la media.
Lo z-score di un valore rispetto a una distribuzione si calcola come

$$z = \frac{X - \mu}{\sigma} \tag{4.3}$$

Dove z è lo score, X è il valore del numero, μ è il valore medio della distribuzione e σ la deviazione standard.

Lo z-score può quindi venir utilizzato per trasformare gli elementi di una distribuzione a seconda del loro valore rispetto alla media della distribuzione stessa. A seguito della trasformazione infatti, il 99% dei valori ottenuti rimane nel range compreso tra -3 e 3. Valori negativi stanno a significare che l'elemento è più piccolo della media, viceversa valori positivi indicano elementi maggiori della media. Più un elemento si allontana dallo 0, più questi si discosterà dalla media della distribuzione, mentre più un elemento si avvicina allo 0, che rappresenta la media esatta, più avrà valori di partenza prossimi alla media di tutta la distribuzione. Questo può essere utile in quanto le distanze tra i valori vengono amplificate, garantendo nel caso particolare dello spazio distribuzionale considerato, di rappresentare efficacemente il distacco tra elementi con valori particolarmente elevati per un'emozione rispetto ad altri con valori più vicini alla norma. Si è deciso di provare dunque a effettuare i medesimi esperimenti anche considerando questo tipo di normalizzazione.

Per quanto riguarda l'aspetto tecnico, la normalizzazione è stata effettuata per ognuna delle emozioni, andando via via a considerare ognuno dei valori assunti da ogni parola come facenti parte della distribuzione. A differenza del caso precedente, in cui la normalizzazione è stata effettuata considerando come unità base la parola, e all'interno di essa ognuno dei valori differenti per le varie emozioni, qui si è considerato l'emozione in generale, andando a modificare via via i valori delle singole parole rispetto alla distribuzione generale dei valori per l'emozione considerata. Si è utilizzato il modulo *stats* della libreria Python *scipy*, che contiene al suo interno una formula automatica per il calcolo dello z-score data una distribuzione. Di seguito è

riportata parte del codice utilizzato, che esegue l'operazione richiesta per tutti e tre i lessici emotivi:

```
from scipy import stats
[...]
for emolex in emolexes:
   for emotion in emotions:
      values = []
      for word, value in emolex[emotion].iteritems():
         values.append(float(value))


      array = np.array(values)
      zscore_values = stats.zscore(array)
      i = 0
      for word, value in emolex[emotion].iteritems():
         emolex[emotion][word] = zscore_values[i]
         i+=1
```

I valori sono stati successivamente scritti su file da poter utilizzare direttamente nello script di estrazione delle feature.

Per portare un esempio della modifica dei valori a seguito della normalizzazione, sono riportate in seguito alcune delle parole con gli score più alti per i nomi l'emozione GIOIA, prima e dopo.

Una volta effettuata la normalizzazione, si è proceduto come nei casi precedenti alla creazione di vari modelli, al fine di individuare i migliori in termini di prestazioni sul training-set. Come per la normalizzazione per parola, anche in questo caso non sono stati ripetuti tutti gli esperimenti, ma solamente quelli ritenuti più promettenti osservando i risultati ottenuti dal modello non normalizzato. Anche in questo caso, i risultati completi sono disponibili in appendice, mentre verranno di seguito esposti

Emotion	Word	Value Std.	z-score
Gioia	gioia-s	0.769305	6.032448
Gioia	trionfo-s	0.648426	4.677928
Gioia	soddisfazione-s	0.642233	4.608532
Gioia	felicità-s	0.636774	4.547361
Gioia	serenità-s	0.612624	4.276746

Tabella 4.8: Valori più alti per i nomi dell'emozione GIOIA, prima e dopo la normalizzazione con z-score

i tre modelli migliori soprattutto in base al valore di recall della classe 1 del task POS.

All Features Di nuovo uno dei modelli migliori è risultato essere quello creato tramite l'utilizzo di tutte le feature, sebbene il valore per la recall di POS:1 non sia tra i più alti riscontrati.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.74	0.68	0.71
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.9: Risultati del training-set sul modello normalizzato con z-score All Features

No Top Diversamente dagli altri gruppi di modelli, grazie alla normalizzazione tramite z-score è emerso un altro modello con valori interessanti. In questo caso infatti, a essere rimosse sono state le feature contenenti i valori massimi per ogni emozione, chiamate TOP. I valori sono interessanti in quanto sembrano in linea con i modelli migliori. Si potrebbe ipotizzare che questo risultato possa essere portato proprio dalla modifica dei valori emotivi, molto più vari e potenzialmente fuorvianti

per il classificatore. Se difatti il massimo per ogni emozione prima della normalizzazione poteva essere tra 0.9 e 1, dopo si riscontrano valori anche superiori al 5 in casi isolati ma comunque presenti. Per quanto comuque il miglioramento sia abbastanza

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.93	0.85
	1	0.66	0.36	0.46
NEG	0	0.74	0.67	0.7
	1	0.61	0.69	0.65
GLOBAL		0.71	0.71	0.71

Tabella 4.10: Risultati del training-set sul modello normalizzato con z-score No Top

netto, questo avviene solamente per la classe 1 del task POS, mentre in altri casi non è evidente, o addirittura i valori risultano essere più bassi.

No Score Anche utilizzando la normalizzazione tramite z-score, uno dei modelli migliori risulta ancora una volta essere quello senza lo score di distintività, come mostrato in tabella.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.93	0.85
	1	0.66	0.36	0.46
NEG	0	0.74	0.67	0.7
	1	0.61	0.69	0.65
GLOBAL		0.71	0.71	0.71

Tabella 4.11: Risultati del training-set sul modello normalizzato con z-score No Score

No Distinctive Infine, è stato testato anche il comportamento del modello in caso di rimozione delle feature distintive. Anche utilizzando la nuova normalizzazione si è notato un risultato comunque interessante, vista la rimozione di un numero

relativamente alto di feature sul totale. Il calo difatti è apprezzabile solo per quanto riguarda la classe 1 di POS, ma risulta comunque non estremamente significativo.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.74	0.68	0.71
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.12: Risultati del training-set sul modello normalizzato con z-score No Distinctive

4.5.4 Aggiunta della polarità

Gli esperimenti finora non utilizzavano i valori di polarità calcolati per ognuna delle parole emotive riscontrate all'interno dei tweet.

Dati i risultati promettenti degli esperimenti svolti in precedenza, si è deciso di non ripetere tutti gli esperimenti prodotti, ma semplicemente di evidenziare i risultati su alcuni degli esperimenti migliori, al fine di comprendere in che modo l'aggiunta di questa feature potesse portare un miglioramento.

Sono state quindi nuovamente testate alcune configurazioni. Anche in questo caso, per semplicità ci si limiterà a presentare solamente le migliori individuate in fase di addestramento e sviluppo, mentre le altre possono essere trovate in appendice.
I modelli migliori in questo caso sono risultati essere quattro.

All Features Il primo modello presentato è ancora una volta quello contenente tutte le feature estratte, oltre anche alle feature presenti nel modello baseline.
Le feature utilizzate sono quindi tutte quelle presentate in 4.5.1, con l'aggiunta delle polarità.

I risultati ottenuti sul training set da questo modello sono presentati in tabella. Il modello è risultato essere il migliore in assoluto, tra quelli facenti utilizzo dello spazio emotivo di base, come mostra la tabella.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.63	0.36	0.46
NEG	0	0.73	0.69	0.71
	1	0.62	0.67	0.64
GLOBAL		0.71	0.72	0.71

Tabella 4.13: Risultati del training-set sul modello All Features+polarità

No Score Successivamente, è stato testato anche il modello senza la feature di score. I valori sono molto simili al modello con tutte le feature, come mostrato in tabella.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.63	0.36	0.46
NEG	0	0.73	0.69	0.71
	1	0.62	0.67	0.64
GLOBAL		0.71	0.72	0.71

Tabella 4.14: Risultati del training-set sul modello No Score+polarità

No Distinctive Ancora una volta anche il modello senza feature di distintività è risultato comunque avere valori buoni, sebbene leggermente inferiori a quelli presentati per il modello con tutte le feature. Tuttavia si nota nuovamente un leggero decremento per quanto riguarda la recall per la classe 1 del task POS, che come già detto è stato un punto fermo per valutare le prestazioni del modello.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.73	0.69	0.71
	1	0.62	0.67	0.65
GLOBAL		0.71	0.72	0.71

Tabella 4.15: Risultati del training-set sul modello No Distinctive+polarità

No Score + No Distinctive Infine questo ultimo modello si è deciso di unire l'assenza di Score emotivo con l'assenza delle feature riguardanti la distintività. Anche in questo caso, si denota un leggero calo delle prestazioni, per il quale va comunque tenuto conto dell'assenza di 33 feature, come nel caso precedente. Inoltre, si nota che il calo di prestazioni avviene proprio nell'area di interesse, ovvero per la classe 1 del task POS.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.35	0.44
NEG	0	0.73	0.69	0.71
	1	0.62	0.67	0.64
GLOBAL		0.71	0.71	0.70

Tabella 4.16: Risultati del training-set sul modello No Score + No Distinctive

Come già evidenziato, per quanto riguarda i modelli prodotti senza alcun tipo di normalizzazione dei i valori presenti nello spazio distribuzionale, il migliore in assoluto è risultato essere comunque quello in cui tutte le feature sono state utilizzate, anche a significare una buona valutazione nella scelta delle stesse.

Al netto degli esperimenti condotti, sicuramente sebbene la normalizzazione, con entrambi i metodi, abbia portato ad alcuni miglioramenti interessanti, non sono risul-

tati particolarmente significativi. Anche per quanto riguarda l'aggiunta delle feature di polarità, sicuramente non si tratta di un incremento di prestazioni fondamentali, tuttavia risulta fondamentale tenerne da conto per gli esperimenti da svolgere con il test-set.

Capitolo 5

Risultati e discussione

Dopo aver esplorato i metodi per preparare i modelli di classificazione, nonché mostrato le prestazioni di questi durante la fase di sviluppo e addestramento del classificatore, risulta infine necessario mostrare e discutere i risultati ottenuti sul test set da una serie di modelli, scelti tra quelli mostrati in precedenza, ritenuti quelli con le caratteristiche più interessanti, e mostratisi più promettenti durante l'addestramento.

In particolare, in questa sezione si vuole mostrare e commentare sette modelli differenti. Gli esperimenti già presentati nel capitolo precedente sono dunque stati ripetuti, al fine di evidenziarne le prestazioni utilizzando il test set. In particolare, sono stati considerati i modelli completi, e quelli senza score emotivo o feature di distintività, sia creati utilizzando lo spazio emotivo standard, sia con la normalizzazione attraverso lo z-score.
Inoltre, successivamente agli esperimenti di base, sono stati prodotti altri modelli, al fine di effettuare un raffronto anche con il modello completo, ovvero CoLingLab Full, prodotto da Passaro et al. (2014)[53], di cui al capitolo 2.4.1, oltre che con il modello considerato come baseline, ovvero il modello in assenza di feature lessicali, detto Shallow Model. Si è scelto di procedere in questo modo per riuscire a evidenziare più efficacemente le differenze tra i modelli prodotti utilizzando il modello lessicale basato sulla risorsa Sentix e i modelli creati a partire dallo spazio distribu-

zionale basato su FBNEWS-2015. Si è cercato inoltre di comprendere così facendo quali fossero le interazioni tra la risorsa lessicale e la risorsa distribuzionale a nostra disposizione.

Per completezza e per facilitare il raffronto si riportano nuovamente i risultati ottenuti dal modello baseline (CoLingLab Shallow Model) e dal modello completo prodotti per il task.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7979	0.7806	0.789
POS	1	0.581	0.4109	0.4814
POS task		0.6893	0.5957	0.6352
NEG	0	0.6923	0.6701	0.681
NEG	1	0.6384	0.5201	0.5732
NEG task		0.6654	0.5951	0.6271
GLOBAL		0.6774	0.5954	0.6312

Tabella 5.1: Risultati sistema CoLingLab

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7578	0.8679	0.8092
POS	1	0.7184	0.2205	0.3374
POS task		0.7381	0.5442	0.5733
NEG	0	0.7369	0.5174	0.608
NEG	1	0.5778	0.6582	0.6154
NEG task		0.6574	0.5878	0.6117
GLOBAL		0.6978	0.566	0.5925

Tabella 5.2: Risultati sistema CoLingLab, Shallow Model

In questo caso la valutazione dei modelli è stata effettuata con gli strumenti forniti da Evalita. Inizialmente anche il set di tweet fornito per il test è stato sottoposto alla stessa fase di pre-processing descritta nel capitolo 4.2, per renderli adeguati all'estrazione delle feature prima e alla classificazione in seguito. Per quanto concerne l'estrazione delle feature, essendo il dataset identico nella forma a quello utilizzato per il training, è stato utilizzato lo stesso script. Successivamente, Weka permette di effettuare la classificazione tramite l'utilizzo di entrambi i set, effettuando l'addestramento sul dataset di training e testandolo successivamente sul test-set.
Una volta completata la fase di classificazione, i risultati sono stati estratti e modificati tramite uno script nella forma richiesta da Evalita per la valutazione, e forniti insieme con la lista di tweet da escludere allo script di Evalita, che restituisce in output le valutazioni sulle classi di ogni task oltre al risultato globale di precision, recall e f-measure. Viene di seguito riportato un esempio della forma dei dati così come prodotti da Weka e a seguito della modifica, oltre al breve script usato per effettuarla.

inst#	actual	predicted	error	prediction	idtw
1	1:10	1:10		0.5	136503827527507968
2	1:10	1:10		0.5	391946157846712320
3	4:01	3:00	+	0.5	153558720008306688
4	1:10	3:00	+	0.5	147281002388135936
5	4:01	4:01		0.5	134995544878759936
6	1:10	3:00	+	0.5	193414126507278337
7	4:01	4:01		0.5	138515518389891072
8	3:00	3:00		0.5	194091628485550080
9	3:00	4:01	+	0.5	149251704431456256

Prima

136503827527507968	0	1	0	0	1
391946157846712320	0	1	0	0	0
153558720008306688	0	0	0	0	1
147281002388135936	0	0	0	0	1
134995544878759936	0	0	1	0	1
193414126507278337	0	0	0	0	0
138515518389891072	0	0	1	0	1
194091628485550080	0	0	0	0	0
149251704431456256	0	0	1	0	1

Dopo

Figura 5.1: Prima e dopo la regolarizzazione dei dati

```
import csv
from collections import defaultdict
twid_topic = defaultdict(str)
```

```
with open("TEST_DATA.SENTIPOLC Sentiment Polarity Classification
    - Evalita 2014.csv") as infile:
   next(infile)
   for i, line in enumerate(infile):
      splitted = line.split(",")
      idtwitter = splitted[0].strip('"')
      top = splitted[5].strip('"')
      twid_topic[idtwitter] = top
      #idtwitter,subj,pos,neg,iro,top,TEXT = splitted
with open("results_1.csv", "r") as infile,
    open("results_1_clean.csv", "wb") as outfile:
   wr = csv.writer(outfile, quoting=csv.QUOTE_ALL)
   next(infile)
   for line in infile:
      splitted = line.split(",")
      inst,actual,predicted,error,prediction,idtw = splitted
      split_predicted = predicted.split(":")
      idtw = idtw.strip("\n")
      pos = split_predicted[1][0]
      neg = split_predicted[1][1]
      topic = twid_topic[idtw]
      write = [idtw, "0", pos, neg, "0", topic]
      wr.writerow(write)
```

Nessuno dei modelli prodotti è risultato avere la predominanza assoluta in ognuno dei campi. Si denota infatti al contrario un certo equilibrio nelle prestazioni tra i vari modelli. Alcuni risultano avere migliori prestazioni di altri ma limitatamente a certi valori. Soltanto in un caso si è evidenziata la predominanza quasi assoluta nelle prestazioni di un modello rispetto ad un altro, in particolare per quanto riguarda i modelli 5 e 6, sviluppati tenendo come base di partenza a cui aggiungere le feature emotive il modello con Sentix (CoLingLab Full) sviluppato da Passaro et al.

(2104)[53] anziché la baseline rappresentata dal modello Shallow.

Saranno quindi ora riportati e discussi nelle sezioni successive i risultati ottenuti dai vari modelli applicati al test set. Ogni modello sarà discusso singolarmente, ma la lista sarà ordinata in macro-sezioni volte a suddividere i termini di paragone in quanto al modello di riferimento usato per valutare eventuali incrementi di prestazione oltre che i diversi spazi emotivi utilizzati.

5.1 Modelli senza Sentix

Per prima cosa, verranno analizzati i modelli che rappresentano il focus reale della presente ricerca. Si tratta dei modelli sviluppati a partire dal modello Shallow, considerato come baseline per gli esperimenti. Caratteristica primaria di questo modello è quella di non considerare le feature lessicali prodotte attraverso Sentix. Risulta quindi una base di partenza comunque solida, soprattutto in relazione all'esiguità di feature presenti. Sono evidenziati in grassetto all'interno delle tabelle con i risultati quelli che superano gli altri modelli (limitatamente al confronto con la baseline).

I primi due modelli presentati sono stati creati tramite l'utilizzo dello spazio emotivo standard, senza normalizzazioni di alcun tipo. In particolare, sono stati considerati i due modelli con le prestazioni generalmente migliori tra quelle mostrate nel capitolo precedente attraverso la valutazione sul training set. Questi sono risultati essere il modello Senza score emotivo, dal quale è stato escluso lo score di emotività, e il modello che considera tutte le feature prodotte per la classificazione.

5.1.1 Modello 1: Shallow + DistrFBNEWS(NoScore)

Il primo modello presentato è appunto quello che utilizza tutte le feature a eccezione dello score di emotività. Si è infatti notato, in fase di addestramento, come l'ablazione di questa feature potesse risultare potenzialmente efficace per produrre

risultati più interessanti. I risultati del modello applicato al test set sono presentati in tabella.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	**0.7842**	0.8291	0.806
POS	1	0.6531	**0.3386**	**0.446**
POS task				**0.626**
NEG	0	0.7331	0.6447	0.6861
NEG	1	0.6471	0.5972	0.6212
NEG task				0.6536
GLOBAL				0.6398

Tabella 5.3: Risultati modello Standard DistrFBNEWS(NoScore)

5.1.2 Modello 2: Shallow+ DistrFBNEWS(Full)

Il secondo modello prodotto implementa invece tutte le feature per la classificazione, senza eliminarne nessuna. I risultati sono presentati in tabella.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7834	0.8305	0.8063
POS	1	0.6552	0.3351	0.4434
POS task				0.6248
NEG	0	**0.735**	**0.6456**	**0.6874**
NEG	1	**0.6488**	**0.5995**	**0.6232**
NEG task				**0.6553**
GLOBAL				**0.6401**

Tabella 5.4: Risultati modello Standard DistrFBNEWS

Sembra abbastanza evidente come, oltre alla miglior prestazione in generale per quanto riguarda il task NEG e il valore di f-score globale, che lo rendono all'atto

pratico il miglior modello prodotto tra quelli non facenti utilizzo di feature lessicali estratte da Sentix, tra i due modelli prodotti utilizzando lo spazio distribuzionale emotivo standard senza normalizzazioni ha prestazioni migliori in quasi tutte le misure per i vari task. Gli unici campi in cui non è superiore al modello 1 sono la precision per POS:0, e recall e f-measure per POS:1. Anche il valore globale per il task POS propende per il modello 1. Tuttavia, gli scarti tra i due sono in questi casi estremamente limitati.
Tra i due modelli visti fin'ora dunque, sebbene gli score per quanto riguarda il task POS siano sbilanciati verso il Modello 1, sembra comunque prevalere a livello di prestazioni il Modello 2, facente utilizzo di tutte le feature a disposizione estratte dallo spazio distribuzionale. Difatti, anche per il calcolo della f-measure globale questi risulta superiore. È comunque importante precisare ancora una volta come nella quasi totalità dei casi le prestazioni dei due modelli non differiscano enormemente le une dalle altre.

5.1.3 Modello 3: Shallow + Z-DistrFBNEWS(NoScore)

Il secondo gruppo di modelli presentati prevede ancora una volta l'utilizzo del modello CoLingLab Shallow come baseline. In questo caso tuttavia, i modelli sono riferiti allo spazio distribuzionale normalizzato attraverso lo z-score. Si è deciso di valutare anche in questo caso alcuni dei modelli migliori.

Per la produzione del terzo modello è stata esclusa nuovamente la feature di scoring dell'emotività nel tweet. I risultati ottenuti, sebbene inferiori a tutti gli altri modello prodotti, sembrano comunque meritevoli di interesse in quanto lo scarto anche in questo caso risulta minimo rispetto agli altri.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7809	0.8291	0.8043
POS	1	0.6458	0.338	0.4351
POS task				0.6197
NEG	0	0.7276	0.6418	0.682
NEG	1	0.6421	0.5903	0.6151
NEG task				0.6486
GLOBAL				0.6341

Tabella 5.5: Risultati modello Z-Normalized No Score

5.1.4 Modello 4: Shallow + Z-DistrFBNEWS(NoScoreNoDistinct)

I risultati prodotti eliminando anche le feature riguandanti la distinitivtà emotiva delle parole riscontrate hanno prodotto valori certamente più elevati, soprattutto per quanto riguarda alcune valutazioni sul task POS, per i quali è risultato essere il migliore tra i modelli prodotti a partire dal modello CoLingLab Shallow. Anche rispetto ai valori riscontrati per il modello 3, in cui è stata esclusa solo la feature di score, si osserva una condizione di prevalenza su tutti i campi. Ciononostante, anche in questo caso lo scarto tra i modelli è decisamente marginale.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7835	**0.8313**	**0.8067**
POS	1	**0.6574**	0.3351	0.4439
POS task				0.6253
NEG	0	0.7289	0.6437	0.6837
NEG	1	0.6441	0.5915	0.6167
NEG task				0.6502
GLOBAL				0.6377

Tabella 5.6: Risultati modello Z-Normalized No Score, No Distinctive

Come si è visto dunque dallo scarto tra tutti i modelli finora presentati, riferiti alla baseline del modello CoLingLab Shallow, hanno tutti evidenziato innanzitutto risultati relativamente simili, discordanti al massimo di qualche punto l'uno dall'altro. Inoltre, non è stato possibile evidenziare un modello anche solo leggermente superiore per prestazioni su tutti i campi rispetto agli altri. In tabella 5.8 si mostra infatti come ogni modello, ad eccezione del numero 3, abbia mostrato le migliori prestazioni in alcuni campi a discapito degli altri. Se difatti il Modello Shalllow+DistrFBNEWS(full)

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	St NoScore	Z NoScoreNoDist	Z NoScoreNoDist
POS	1	Z NoScoreNoDist	St NoScore	St NoScore
POS task				St No Score
NEG	0	St AllFeat	St AllFeat	St AllFeat
NEG	1	St AllFeat	St AllFeat	St AllFeat
NEG task				St AllFeat
GLOBAL				St AllFeat

Tabella 5.7: Miglior risultato per ogni task sui quattro modelli presi in considerazione

è riuscito a ottenere i migliori risultati in assoluto per ognuna delle misurazioni del task NEG, e ha ottenuto il valore massimo per la prestazione calcolato tramite la f-measure globale, ciò non è vero per il task POS, in cui si sono contesi i risultati migliori il Modello 1 e il Modello 4. Il primo ottiene i valori migliori per precision di POS:0 e recall e f-measure di POS:1, mentre l'altro per la precision di POS:1 e recall e f-measure di POS:0. Il valore globale per il task POS in questo caso è massimo per il Modello 1.

Dai dati presentati si evince dunque come non sia possibile stabilire, almeno con questo tipo di feature, un modello assolutamente migliore degli altri, complice anche lo scarto esiguo tra i risultati. Basti osservare che il valore massimo per GLOBAL, raggiunto dal Modello 2 con 0.6401, è separato dal peggiore (Z-DistrFBNEWS(NoScoreNoDistinct)) di appena 0.006 punti. Chiaramente dunque

la differenza di prestazioni, anche in contesti d'utilizzo reali, risulterebbe essere di scarso interesse.

Tuttavia di grande interesse è soprattutto il confronto con il modello di base da cui si è partiti per la costruzione del classificatore. Si è in questo caso proceduto con l'analisi di uno dei quattro modelli a confronto con il modello Shallow prodotto dal CoLingLab. Si è deciso di considerare come modello d'esempio Shallow+DistrFBNEWS(Full), in quanto risulta essere quello con f-score globale migliore, oltre ad aver raggiunto il punteggio massimo per più della metà delle misure prese in considerazione.

In tabella è mostrato il risultato ottenuto dal modello, a cui sono stati sottratti i valori di CoLingLab Shallow, in modo da mostrare lo scarto tra i due. Gli scarti negativi rappresentano le aree in cui il modello baseline è risultato superiore a quello prodotto. Dalla tabella si evince chiaramente come in buona parte delle misure pre-

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	+0.0256	-0.0374	-0.0029
POS	1	-0.0632	+0.1146	+0.106
POS task				+0.0515
NEG	0	-0.0019	+0.1282	+0.0794
NEG	1	+0.071	-0.0587	+0.0078
NEG task				+0.0436
GLOBAL				+0.0476

Tabella 5.8: Scarto tra i risultati di DistrFBNEWS(Full)e CoLingLab Shallow

se in considerazione il modello prodotto si dimostri superiore, anche di diversi punti percentuali, rispetto alla baseline. Di particolare interesse come già evidenziato è risultata essere la recall di POS:1, che potrebbe essere definita come il tallone d'Achille del modello Shallow. In questo caso, il valore è stato elevato di ben 11 punti rispetto alla baseline, un risultato senza dubbio confortante. Un simile livello di incremento

è avvenuto anche per la recall di NEG:0. I campi in cui il modello presentato è risultato inferiore inoltre mostrano degli scarti generalmente ben al di sotto degli incrementi evidenziati. Infine, anche il valore globale di f-score calcolato da Evalita ha mostrato un incremento di ben 4 punti. Un risultato senza dubbio interessante, in quanto in una ipotetica competizione con lo stesso metro di valutazione il modello prodotto sarebbe stato valutato come decisamente superiore al modello Shallow.

5.2 Modelli con Sentix

Dopo aver evidenziato le caratteristiche riguardanti i modelli fondamentali per la presente ricerca, si è deciso di effettuare alcuni esperimenti aggiuntivi, volti a mostrare le prestazioni di alcuni modelli sviluppati tenendo in considerazione come base di partenza il modello Full prodotto dal CoLingLab. In questo caso, sono state utilizzate in aggiunta al modello anche lessicale basato su Sentix le medesime feature emotive prodotte per gli altri modelli, in congiunzione con lo spazio emotivo normalizzato tramite z-score, al fine di mostrare eventuali miglioramenti nelle prestazioni anche del modello completo attraverso l'uso delle nuove feature introdotte. Ciò risulta interessante anche dal punto di vista tecnico e pratico oltre che teorico, in quanto permette di comprendere meglio quanto un modello distribuzionale di natura emotiva possano incidere sulla classificazione, in particolar modo se utilizzate all'interno di un classificatore con un numero esponenzialmente maggiore di esse. Difatti, se per i modelli sviluppati tenendo conto della baseline le feature sono circa 200, in questo caso risultano essere in numero ben superiore al migliaio, delle quali molte sono di natura lessicale. Per una spiegazione completa del modello CoLingLab Full e delle sue feature si rimanda a Passaro et al. (2014)[53].

5.2.1 Modello 5: Full+Z-DistrFBNEWS(NoScore)

Il primo modello completo è stato prodotto rimuovendo ancora una volta la feature di scoring emotivo, individuata, come mostrato nel capitolo precedente, come potenzialmente dannosa alla classificazione. Pur non essendo il migliore dei due modelli,

merita comunque la presentazione dei risultati, anche per un raffronto con il modello successivo, sicuramente più efficace. Le feature utilizzate sono dunque tutte le feature del modello CoLingLab Full oltre a tutte le feature (tranne lo score) emotive prodotte.
I risultati sono riportati in tabella. Come si può osservare, anche comparando i dati

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.8002	0.7667	0.7831
POS	1	0.5641	0.4268	0.4859
POS task				0.6345
NEG	0	0.7162	0.6682	0.6914
NEG	1	0.6537	**0.5604**	0.6035
NEG task				0.6474
GLOBAL				0.641

Tabella 5.9: Risultati modello Full+Z-DistrFBNEWS(NoScore)

con quelli prodotti per il modello full, si nota un leggero incremento di prestazioni soprattutto per quanto riguarda il task NEG. Difatti questo modello ha ottenuto le migliori prestazioni in assoluto per la recall di NEG:1. A fronte di ciò tuttavia le prestazioni per il task POS sono risultate essere al pari se non inferiori rispetto al modello di partenza, complice anche la già citata sproporzione nei dati di addestramento tra esempi negativi e positivi. Nonostante le prestazioni leggermente inferiori per il task POS tuttavia, è bene notare come lo score di f-measure globale risulti di quasi un punto più alto nel caso dell'aggiunta delle feature emotive, risultato comunque positivo e interessante.

Modello 6: Full+Z-DistrFBNEWS(NoScoreNoDistinct) Il modello successivo, a cui sono state rimosse anche le feature distintive, è risultato sicuramente più efficace del primo, facendo segnare valori più alti per tutti i campi considerati. La tabella riassuntiva mostra ancora una volta come sia stato possibile incrementare le prestazioni del sistema CoLingLab, soprattutto per quanto riguarda il task NEG.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.8059	0.7674	0.7862
POS	1	0.5744	0.4427	0.5
POS task				0.6431
NEG	0	0.717	0.6758	0.6958
NEG	1	0.6599	0.5581	0.6047
NEG task				0.6503
GLOBAL				0.6467

Tabella 5.10: Risultati sistema Full+Z-DistrFBNEWS(NoScoreNoDistinct)

Tuttavia, in questo caso anche molte delle misure effettuate per quanto riguarda il task POS sono risultate superiori al modello di partenza. In casi di prestazioni più basse, lo scarto negativo rimane comunque decisamente ridotto.

Anche nel caso del sistema basato sul modello CoLingLab si è voluta dare quindi una overview della differenza in termini di prestazioni registrate. Si è scelto chiaramente di utilizzare il Modello 6 a fronte delle prestazioni superiori registrate in tutti i campi.

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	+0.083	-0.0132	-0.0028
POS	1	-0.0066	+0.0318	+0.0186
POS task				+0.0079
NEG	0	+0.0247	+0.0057	+0.0148
NEG	1	+0.0215	+0.038	+0.031
NEG task				+0.0232
GLOBAL				+0.0155

Tabella 5.11: Comparativa tra sistema CoLingLab e modello Full+Z-DistrFBNEWS(NoScoreNoDistinct)

Come si può osservare dalla comparativa in tabella si nota ancora una volta un incremento abbastanza consistente soprattutto per il task NEG, a fronte invece di una situazione equilibrata per il task POS. Come già detto, gli incrementi ma anche e soprattutto i decrementi nelle prestazioni risultano estremamente ridotti. Anche in questo caso tuttavia il valore di f-measure globale è stato incrementato di circa un punto e mezzo.

Si potrebbe inoltre precisare che, a differenza dei modelli in cui è stato usato il sistema Shallow come baseline, in questo caso il numero di feature è stato probabilmente di grande impatto nelle prestazioni del classificatore. Difatti è possibile che l'alto numero di feature lessicale utilizzato abbia in qualche modo ridotto le potenzialità delle feature emotive, in numero molto ridotto rispetto alle altre. Le feature emotive infatti ammontano a circa il 10% del totale. Ciononostante anche in questo caso il modello sviluppato è stato in grado di superare, sebbene in misura minore, le prestazioni globali del sistema CoLingLab, rendendolo di fatto migliore in un ipotetico contesto di competizione misurata sul singolo valore globale.

5.3 Modello 7: Z-DistrFBNEWS(NoScore)

Infine, un ultimo test è stato effettuato considerando solamente le feature emotive. Si è deciso di effettuare questo tentativo sia per completezza, sia per poter avere delle informazioni interessanti riguardanti le prestazioni di un classificatore di polarità addestrato solamente attraverso l'utilizzo di feature che rappresentano l'emotività del tweet. Si è cercato in questo modo di capire quanto effettivamente possa essere presente un collegamento tra la polarità di un testo scritto e gli stati emotivi che in esso vengono rappresentati.

Chiaramente, a differenza degli altri modelli presentati, questo ha fatto registrare prestazioni decisamente inferiori, sia per l'esiguo numero di feature, sia per la rilevanza più scarsa delle feature usate per la classificazione. Per il test è stato usato il modello normalizzato con z-score e a cui è stata tolta solo la feature di score emotivo.

Come si può osservare dai risultati prodotti dal test, sebbene il modello non riesca

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.7528	0.8313	0.7901
POS	1	0.5696	0.231	0.3287
POS task				0.5594
NEG	0	0.6732	0.6447	0.6586
NEG	1	0.6064	0.5017	0.5491
NEG task				0.6039
GLOBAL				0.5816

Tabella 5.12: Risultati sistema Solo Z-DistrFBNEWS(NoScore)

a raggiungere i precedenti in termini di prestazioni, i risultati sono comunque molto incoraggianti. Difatti si attestano come in linea con i risultati ottenuti dal sistema CoLingLab Shallow, soprattutto per quanto riguarda il valore di f-measure globale. Vista e considerata la diversità delle feature, e soprattutto la loro appartenenza e produzione all'interno di un contesto emotivo, differente rispetto a quello della classificazione di polarità dei sentimenti, rimane comunque un risultato estremamente interessante, soprattutto perché in grado di riportare dati concreti riguardanti la possibile correlazione tra la polarità di un testo scritto e le emozioni espresse dallo stesso. Le prestazioni non sono infatti consistenti, soprattutto per quanto riguarda uno degli aspetti fondamentali del modello CoLingLab, la recall per il task POS:1, che in questo caso risulta non elevata. Tuttavia è interessante anche notare come il valore di recall per il task POS:0 risulti invece decisamente elevato, e come in generale, sempre in un contesto di ipotetica competizione, sarebbe stato quantomeno equiparabile al modello Shallow e al modello Lexical prodotti da Passaro et al. (2014)[53].

5.4 Discussione dei risultati

Considerando la totalità degli esperimenti svolti si possono effettuare alcune interessanti osservazioni. Innanzitutto, risulta evidente il ruolo delle feature emotive nella classificazione. Questo in particolar modo è avvenuto nel modello 7, solo emotivo, e nei primi 4 modelli, nei quali sicuramente l'emotività ha giocato un ruolo più ampio considerato anche il numero di feature. Difatti, le feature emotive prodotte per gli esperimenti, 157, risultano essere molte di più delle 63 utilizzate nell'esperimento baseline CoLingLab Shallow. Ciononostante, è chiaro come le feature emotive risultino incidere notevolmente nella capacità predittiva del modello. Soprattutto, di grande valore per la ricerca risulta essere la Recall per il task POS:1, che come già detto rimane uno dei punti problematici per il modello Shallow. Grazie all'aggiunta delle feature emotive si è stati in grado di migliorare notevolmente le prestazioni in questo campo, raggiungendo in questo modo uno degli obbiettivi primari della ricerca. Inoltre, anche per quanto riguarda le prestazioni generali si è assistito ad un deciso miglioramento. Difatti, andando a considerare i valori riportati per la f-measure globale, il modello migliore (Modello 2) ha mostrato performance superiori di circa un intero punto rispetto al sistema CoLingLab completo. Questo risultato è particolarmente interessante per alcuni motivi. Innanzitutto, ovviamente il sistema risulta globalmente più efficace, sebbene con un lieve scarto. Nell'ipotetico contesto di una competizione ancora una volta basata sul singolo valore globale di f-measure il sistema prodotto risulterebbe vincitore.

Inoltre il modello riesce a ottenere questo tipo di prestazioni con un numero relativamente esiguo di feature. Questo fattore non è assolutamente da sottovalutare, in quanto è in grado di ridurre i tempi in maniera significativa, sia in fase di addestramento del modello che di classificazione. Difatti, negli esperimenti condotti, il Modello 2 è risultato decisamente più rapido nell'effettuare l'addestramento e la classificazione, con un tempo complessivo di circa 15-20 secondi per il completamento del processo, a fronte del minuto e 30 secondi circa necessari per addestrare e classificare il test set per il sistema CoLingLab. I dati numerici non sono chiaramente assoluti,

in quanto dipendono dalla macchina utilizzata. Tuttavia per entrambi i casi è risultata essere la stessa, con lo stesso quantitativo di risorse assegnate forzatamente all'esecuzione di Weka. È interessante come in ogni caso il sistema con un numero decisamente inferiore di feature possa essere globalmente più efficace.

Analizzando i risultati prodotti dal Modello 6, creato a partire dal sistema CoLingLab completo, si nota invece come i miglioramenti, sebbene presenti in quasi tutte le misure effettuate, fatta eccezione per recall e f-score per POS:0, si mostrino in misura più ridotta. Difatti, i più evidenti risultano essere nell'ordine dei tre punti. Rimane comunque positivo quanto si sia riusciti nuovamente a migliorare soprattutto le prestazioni per quanto riguarda la recall di POS:1. Anche il valore globale, sebbene in misura minore rispetto al rapporto tra Modello 2 e Modello CoLingLab Shallow, è stato incrementato di circa un punto e mezzo. Si è portati a supporre che la ragione principale per questo miglioramento in scala più ridotta sia da ritrovarsi nella relativa esiguità delle feature emotive rispetto alle altre. Se per i modelli prodotti a partire da CoLingLab Shallow le feature emotive ammontavano a buona parte delle feature totali, in questo caso sono proposte al classificatore in misura decisamente più ridotta, con un rapporto di circa 1:10 rispetto al totale delle feature lessicali e non prodotte per il sistema CoLingLab Full. Un numero così ridotto rispetto al totale potrebbe aver condotto l'algoritmo di classificazione a non assegnare il guisto peso alle feature emotive, dovendo prendere in considerazione anche tutte le altre.

Queste considerazioni possono infine portare a una riflessione fondamentale. Da ciò che si è mostrato difatti le prestazioni del sistema prodotto a partire da Sentix come risorsa lessicale e dei sistemi prodotti tramite l'utilizzo di spazi distribuzionali, sono chiaramente equiparabili. Considerando la medesima base di partenza difatti, ovvero il modello CoLingLab Shallow, i risultati per il Modello 2 e per il Sistema CoLingLab Full sono decisamente simili. Questo potrebbe indicare quanto l'utilizzo di risorse distribuzionali come lo spazio emotivo creato possa essere un'ottimo sostituto per risorse lessicali. Tale ipotesi potrebbe garantire alcuni vantaggi.
Innanzitutto, le prestazioni raggiunte dal sistema basato sullo spazio distribuzionale

sono simili, se non superiori, al sistema sviluppato attraverso l'utilizzo di Sentix. La considerazione non è banale, soprattutto andando a evidenziare le diverse caratteristiche delle risorse a disposizione. Se infatti una risorsa lessicale quale Sentix necessita di una fase di preparazione lunga e laboriosa, soprattutto in merito all'annotazione dei termini, al contrario l'utilizzo di spazi distribuzionali permette di sfruttare l'ipotesi distribuzionale per creare una risorsa a partire da un numero estremamente limitato di parole significative, in questo caso estratte con tecniche di crowdsourcing, e successivamente estenderla senza dover ricorrere a nuove annotazioni. Inoltre va considerato che la risorsa Sentix raggiunga nel caso particolare una copertura di circa il 29.5% rispetto ai termini presenti nel dataset di tweet, mentre lo spazio distribuzionale prodotto ha raggiunto una copertura del 57% circa.

Un'altro potenziale vantaggio risulta difatti la possibilità di estendere con grande semplicità uno spazio distribuzionale inserendo al suo interno nuovi termini e ottenendo, senza necessità di nuove annotazioni come per una risorsa lessicale, una copertura sempre più completa. Se difatti per l'ampliamento di una risorsa come Sentix sarebbe necessario ricorrere nuovamente all'annotazione di nuovi termini, attraverso ad esempio tecniche di crowdsourcing, per potenziare lo spazio creato a partire da FBNEWS-2015 sarebbe sufficiente stabilire un nuovo vocabolario, e inserire di conseguenza i termini nello spazio distribuzionale per ottenere anche per questi i valori emotivi desiderati in base alla similarità con i centroidi. Si potrebbe ad esempio proporre di estendere la risorsa proprio andando a considerare le parole non coperte da questa all'interno del dataset di addestramento di un classificatore.

Infine va nuovamente sottolineato come la risorsa prodotta abbia avuto un'ottimo impatto sulla classificazione pur utilizzando un numero molto più ridotto di feature, semplificando in questo modo sia la fase di estrazione delle stesse, sia velocizzando nettamente i tempi di esecuzione a parità di risorse a disposizione.

Capitolo 6

Conclusioni

In conclusione, si può sicuramente affermare di aver raggiunto gli obiettivi prefissati. In primo luogo, è stato creato FBNEWS-2015, una risorsa potenzialmente molto utile ai fini di analisi emotive del testo. La risorsa difatti risulta di dimensioni considerevoli, oltre che, come atteso, con una larga preponderanza di testo emotivo da cui attingere per lo sviluppo di modelli. Si confida che possa essere utilizzata con successo anche in contesti differenti da quello mostrato per la presente tesi. In particolare, risulta sicuramente valida in ambito di emotion detection e task simili.

Secondariamente, si è verificato quanto ipotizzato riguardo la rilevanza di caratteristiche emotive rispetto alla polarità di un testo. Attraverso i modelli prodotti difatti è stato mostrato efficacemente quanto le feature emotive scelte potessero effettivamente incidere sulle prestazioni di un classificatore di polarità, sia da sole che in congiunzione con feature effettivamente estratte allo scopo di individuare la polarità.

Inoltre si è proceduto con successo a rendere più efficace un preesistente modello per il riconoscimento della polarità tramite l'utilizzo di feature emotive. Il modello, che aveva già dimostrato le sue potenzialità nella competizione Evalita 2014, è stato migliorato sia in termini di prestazioni, ovvero f-measure globale, sia in termini di rapidità di esecuzione dell'addestramento e successivamente della classificazione

di nuove istanze. Si è fiduciosi che il sistema prodotto possa essere utilizzato con successo in altre competizioni del medesimo tipo.

Inoltre, si è mostrato come l'utilizzo di risorse basate su spazi distribuzionali rispetto a risorse lessicali più tradizionali possa essere estremamente efficace sia per il singolo task, sia in termini di possibilità di aggiornamento ed estensione, in modo da poter affrontare con successo sfide differenti pur mantenendo la stessa struttura di base.

Ringraziamenti

Desidero ringraziare tutti coloro i quali mi hanno supportato e aiutato durante la stesura di questo lavoro.
Per prima cosa vorrei ringraziare il Professor Alessandro Lenci, perché senza la sua guida questa tesi non esisterebbe.

Un ringraziamento particolare va anche a Lucia Passaro del Laboratorio di Linguistica Computazionale dell'Università di Pisa, che mi ha seguito passo passo durante tutto il progetto, e si è sempre dimostrata disponibile ad ascoltare e aiutare, anche quando non era tenuta a farlo.

Devo infine ringraziare tutte le persone a me più care. Cristina, i miei genitori e tutti i miei amici, in particolare Luca Vullo e Matteo Nocchi che mi sono stati vicini nel momento del bisogno. È a loro che ho dedicato questo traguardo, perché senza il supporto di ognuno di essi non lo avrei mai raggiunto.
Grazie,
Alessandro

Appendices

Appendice A

Crawler Facebook

A.1 Crawler Primario

```
import os
import facebook
import requests
import codecs
import csv
import time
import sys
import dateutil.parser as dateparser
import configuration as conf
import functions as foo

#recupera l'access token
access_token = conf.access_token
#legge la lista degli utenti con (eventuale) relativo ultimo
    timestamp
with codecs.open("user_list_facebook.csv", "r+",
    encoding="utf-8") as user_f:
   reader = csv.reader(user_f)
   user_list = {rows[0]:rows[1] for rows in reader}
```

```
#salva il momento corrente
now = str(time.time())
#apre il file per il corpus, il file per l'indice degli id e il
    file per salvare eventuali errori
with codecs.open("data/corpus_"+now+".txt","w+",
    encoding="utf-8") as f,
    codecs.open("data/corpus_index_"+now+".csv", "w+",
    encoding="utf-8") as csvfile, codecs.open("missing_log.csv",
    "a", encoding="utf-8") as logfile:
   fieldnames = ['id', 'parent_post_id', 'parent_comment_id',
       'type', 'story']
   writer_csv = csv.DictWriter(csvfile, fieldnames=fieldnames,
       restval="")
   writer_csv.writeheader()
   start_time = time.ctime()
   start = time.time()
   f.write('\n')
   #variabili che effettuano il conteggio di post e commenti
       scaricati
   counter_posts = conf.counter_posts
   counter_comments = conf.counter_comments
   inittimestamp = conf.inittimestamp
   #iterazione per ogni utente
   for user in user_list:
      #aggiorna i valori per l'utente corrente
      conf.update = True
      current_user = user
      str_oldest_timestamp = user_list.get(user)
      until = '.until('+str_oldest_timestamp+')'
      since = '.since('+inittimestamp+')'
      #argomenti della chiamata, effettuata con since se presente
          valore nella userlist, altrimenti vuota
```

```
if (str_oldest_timestamp != ""):
   args = {'fields' : 'posts'+since+until+'{message, story,
       created_time, type, name, likes.summary(true), shares,
       comments.summary(true){message,
       likes.summary(true),comments.summary(true){message,
       likes.summary(true),comments.summary(true),
       created_time}, created_time}}', }
   oldest_timestamp=int(user_list.get(user))
else:
   args = {'fields' : 'posts'+since+'{message, story,
       created_time, type, name, likes.summary(true), shares,
       comments.summary(true){message,
       likes.summary(true),comments.summary(true){message,
       likes.summary(true),comments.summary(true),
       created_time}, created_time}}', }
   oldest_timestamp = 0
try:
   graph = facebook.GraphAPI(access_token=access_token)
   profile = graph.get_object(current_user)
   retrieved_data = graph.get_object(profile['id'], **args)
except Exception,e:
   print 'exception encountered while retrieving data: ',
       str(e)
   print 'saving status...'
   with codecs.open("user_list_facebook.csv", "w+",
       encoding="utf-8") as user_f_write:
      w = csv.writer(user_f_write, delimiter=",")
      for user in user_list:
         w.writerow([user, user_list[user]])
   print 'saved'
   print 'quitting...'
   quit()
```

```
#recupero post
posts = retrieved_data['posts']
current_timestamp=0
while True:
   try:
      for post in posts['data']:
         foo.getInfo(post, user_list, current_user,
             current_timestamp, oldest_timestamp, f,
             writer_csv)
      posts = requests.get(posts['paging']['next']).json()
   except Exception,e:
      #se lancia un'eccezione del grafo allora ci sono stati
          problemi
      if (str(facebook.GraphAPIError(e).result) !=
          "'paging'")&(str(facebook.GraphAPIError(e).result)
          != "'data'"):
         print facebook.GraphAPIError(e).result
         if((current_timestamp-oldest_timestamp>0)&(oldest_timestamp!=0)):
            logwriter =csv.writer(logfile, delimiter=",")
            logwriter.writerow([current_user,
                current_timestamp, oldest_timestamp])
         #aspetta 35 minuti per sicurezza prima di riprendere
             con il nuovo utente
         break
      else:
         #controlla per sicurezza di non essersi lasciato
             indietro post anche in questo caso
         if((current_timestamp-oldest_timestamp>0)&(oldest_timestamp!=0)):
            logwriter =csv.writer(logfile, delimiter=",")
            logwriter.writerow([current_user,
                current_timestamp, oldest_timestamp])
         print current_user, "finished"
```

```
            #non serve aspettare
            break
end_time = time.ctime()
with codecs.open("user_list_facebook.csv", "w+",
    encoding="utf-8") as user_f_write:
   w = csv.writer(user_f_write, delimiter=",")
   for user in user_list:
      w.writerow([user, user_list[user]])
#print "oldest: ", time.ctime(int(user_list[user]))
print ("--- %s seconds ---" % (time.time()- start))
```

A.2 Funzioni utilizzate

```
import facebook
import requests
import codecs
import csv
import time
import sys
import dateutil.parser as dateparser
import configuration as conf
import re

def generateNewToken():
   app_id = conf.app_id
   app_secret = conf.app_secret
   acces_token = facebook.get_app_access_token(app_id, app_secret)
   return acces_token
def getInfoComment(writefile, csvfile_writer, current_user,
    comment, post, parent=None):
   parsed_date=dateparser.parse(comment['created_time'])
```

```
parsed_date_unix= int(time.mktime(parsed_date.timetuple()))
identifier = comment['id']
parent_post = post['id']
if not parent:
   parent_comment = ""
else:
   parent_comment = parent['id']
creation_time = str(parsed_date_unix)
likes = comment['likes']
like_count = str(likes['summary']['total_count'])
try:
   replies = comment['comments']
   replies_count = str(replies['summary']['total_count'])
except:
   replies_count = ""
try:
   message = comment['message']
except:
   message = ""
try:
   link_name = str('['+comment['name']+']')
except:
   link_name = ""
writefile.write('<doc user="'+current_user+'"
    id="'+identifier+'" type="comment"
    parent_post="'+parent_post+'"
    parent_comment="'+parent_comment+'"
    date="'+creation_time+'" likes="'+like_count+'"
    comments="'+replies_count+'">\n\n')
writefile.write(message+link_name+'\n\n')
writefile.write('</doc>\n\n')
str_id = str(identifier)
```

```
    str_parent_post = str(parent_post)
    str_parent_comment = str(parent_comment)
    if (str_parent_comment == str_parent_post):
       csvfile_writer.writerow({'id': str_id, 'parent_post_id':
           str_parent_post})
    else:
       csvfile_writer.writerow({'id': str_id, 'parent_post_id':
           str_parent_post, 'parent_comment_id':
           str_parent_comment})
    conf.counter_comments+=1
    if (replies_count > 0):
       for reply in replies['data']:
          getInfoComment(writefile, csvfile_writer, current_user,
              comment=reply, post=post, parent=comment)
def getInfo(post, user_list, current_user, current_timestamp,
    oldest_timestamp, writefile, csvfile_writer):
    parsed_date=dateparser.parse(post['created_time'])
    parsed_date_unix= int(time.mktime(parsed_date.timetuple()))
    user_list[current_user]=parsed_date_unix
    identifier = post['id']
    creation_time = str(parsed_date_unix)
    try:
       location = post['place']
    except:
       location = ""
    likes = post['likes']
    like_count = str(likes['summary']['total_count'])
    comments = post['comments']
    comments_count = str(comments['summary']['total_count'])
    try:
       shares = post['shares']
       shares_count = shares['count']
```

```
except:
   shares_count = ""
try:
   message = post['message']
except:
   message = ""
try:
   link_name = str('['+post['name']+']')
except:
   link_name = ""
current_timestamp = parsed_date_unix
writefile.write('<doc user="'+current_user+'"
    id="'+identifier+'" type="post" parent_post=""
    parent_comment="" date="'+creation_time+'"
    location="'+location+'" likes="'+like_count+'"
    comments="'+comments_count+'"
    shares="'+str(shares_count)+'">\n\n')
writefile.write(message+link_name+'\n\n')
writefile.write('</doc>\n\n')
#create strings for the variables to write on support CSV table
str_id = str(identifier)
str_type = str(post['type'])
try:
   str_story = str(post['story'])
except:
   str_story = ""
#write information on CSV
csvfile_writer.writerow({'id': str_id, 'type': str_type,
    'story': str_story})
conf.counter_posts+=1
try:
   comments = post['comments']
```

```
      while True:
         try:
            #bisogna lavorare QUI per avere commenti e replies!!!!
            for comment in comments['data']:
               getInfoComment(writefile, csvfile_writer,
                   current_user, comment=comment, post=post)
            comments =
                requests.get(comments['paging']['next']).json()
         except KeyError:
            break
   except:
   pass

def getInfoError(post, current_user, current_timestamp,
    oldest_timestamp, writefile, csvfile_writer):
   print 'current timestamp: ',current_timestamp
   parsed_date=dateparser.parse(post['created_time'])
   parsed_date_unix= int(time.mktime(parsed_date.timetuple()))
   identifier = post['id']
   creation_time = str(parsed_date_unix)
   try:
      location = post['place']
   except:
      location = ""
   likes = post['likes']
   like_count = str(likes['summary']['total_count'])
   comments = post['comments']
   comments_count = str(comments['summary']['total_count'])
   try:
      shares = post['shares']
      shares_count = shares['count']
   except:
```

```
   shares_count = ""
try:
   message = post['message']
except:
   message = ""
   print message
try:
   link_name = str('['+post['name']+']')
except:
   link_name = ""
print parsed_date_unix - oldest_timestamp
if ((parsed_date_unix-oldest_timestamp)>0):
   print 'da stampare'
   conf.current_timestamp = parsed_date_unix
   writefile.write('<doc user="'+current_user+'"
       id="'+identifier+'" type="post" parent_post=""
       parent_comment="" date="'+creation_time+'"
       location="'+location+'" likes="'+like_count+'"
       comments="'+comments_count+'"
       shares="'+str(shares_count)+'">\n')
   writefile.write(message+link_name+'\n\n')
   writefile.write('</doc>\n')
   #create strings for the variables to write on support CSV
       table
   str_id = str(identifier)
   str_type = str(post['type'])
   try:
      str_story = str(post['story'])
   except:
      str_story = ""
   #write information on CSV
```

```
            csvfile_writer.writerow({'id': str_id, 'type': str_type,
                'story': str_story})
            conf.counter_posts+=1
            try:
               comments = post['comments']
               while True:
                  try:
                  #bisogna lavorare QUI per avere commenti e replies!!!!
                     for comment in comments['data']:
                        getInfoComment(writefile, csvfile_writer,
                            current_user, comment=comment, post=post)
                     comments =
                         requests.get(comments['paging']['next']).json()
                  except KeyError:
                     break
            except:
               pass
         elif ((parsed_date_unix-oldest_timestamp)==0):
            conf.current_timestamp = parsed_date_unix
            pass
         else:
            pass
         print 'timestamp difference: ',
             (current_timestamp-oldest_timestamp)
```

Appendice B

Estrattore di feature emotive

```
import collections
import codecs
from collections import defaultdict, Counter
import numpy
from numpy import cumsum
from cdecimal import Decimal
import operator
import csv
import cStringIO
import pickle
import json

content_POS = ["S" ,"A", "V", "B"]
emotions = ['rabbia', 'paura', 'tristezza', 'gioia', 'attese',
    'disgusto', 'sorpresa', 'fiducia']

emolex_all = defaultdict(lambda: defaultdict(list))
emolex_S = defaultdict(lambda: defaultdict(list))
emolex_A = defaultdict(lambda: defaultdict(list))
emolex_V = defaultdict(lambda: defaultdict(list))
emo_list_unsorted = []
```

```
with
    codecs.open("FB_space/corpus_FB_2015.cleaned.fixed.sorted.ppmi.S.cos",
    "rb", encoding="utf-8") as emo_space_S,
    codecs.open("FB_space/corpus_FB_2015.cleaned.fixed.sorted.ppmi.A.cos",
    "rb", encoding="utf-8") as emo_space_A,
    codecs.open("FB_space/corpus_FB_2015.cleaned.fixed.sorted.ppmi.V.cos",
    "rb", encoding="utf-8") as emo_space_V:
   emo_list_unsorted = []
   files = [emo_space_S, emo_space_A, emo_space_V]
   for file in files:
      for line in file:
         line.strip()
         splitted = line.split("\t")
         emotion = splitted[0]
         word = splitted[1]
         value = splitted[2].strip()
         emolex_all[word][emotion] = unicode(value)
         if file ==emo_space_S:
            emolex_S[word][emotion] = unicode(value)
         elif file == emo_space_A:
            emolex_A[word][emotion] = unicode(value)
         elif file == emo_space_V:
            emolex_V[word][emotion] = unicode(value)
         emo_list_unsorted.append([emotion, word, value])
emo_list_sorted_emotion = sorted(emo_list_unsorted, key=lambda
    element: (element[2]), reverse=True)
emo_list_tot = sorted(emo_list_sorted_emotion, key=lambda
    element: (element[0]))

with codecs.open("emotive_words.fixed", "r", encoding="utf-8") as
    ITEM_SEEDS, codecs.open("freq_twitter.txt", "r",
```

```
    encoding="utf-8") as frequencies:
   emo_seeds_freq = []
   freqs = defaultdict(str)
   for line in frequencies:
      line.strip()
      splitted = line.split(" ")
      wordPos, freq = splitted
      wordPos.lower()
      freqs[wordPos] = freq.strip('\n')
   for line in ITEM_SEEDS:
      line.strip()
      splitted = line.split("\t")
      seed,emotion,pos = splitted
      wordPos = (seed+"-"+pos).lower().strip()
      if freqs[wordPos]:
         emo_seeds_freq.append([seed, emotion, pos.strip('\n'),
             int(freqs[wordPos])])
      else:
         emo_seeds_freq.append([seed, emotion, pos.strip('\n'), 0])
   sorted1_emo_seeds_freq = sorted(emo_seeds_freq, key=lambda
       element: (element[3]), reverse = True)
   sorted_emo_seeds_freq = sorted(sorted1_emo_seeds_freq,
       key=lambda element: (element[1]))

emolexes = [emolex_all]
emodist_all = defaultdict(lambda:defaultdict(list))

with
    codecs.open("FB_space_normalized_word_emotion/corpus_FB_2015.cleaned.fixed.s
    "rb", encoding="utf-8") as emo_space_normalized_S,
    codecs.open("FB_space_normalized_word_emotion/corpus_FB_2015.cleaned.fixed.s
    "rb", encoding="utf-8") as emo_space_normalized_A,
```

```
    codecs.open("FB_space_normalized_word_emotion/corpus_FB_2015.cleaned.fixed.s
    "rb", encoding="utf-8") as emo_space_normalized_V:
   emo_list_unsorted = []
   files = [emo_space_normalized_S, emo_space_normalized_A,
       emo_space_normalized_V]
   for file in files:
      for line in file:
         line.strip()
         splitted = line.split("\t")
         emotion = splitted[0]
         word = splitted[1]
         value = splitted[2].strip()
         emodist_all[word][emotion] = unicode(value)

distinctive_emotion_words = []
for word, emos in emodist_all.iteritems():
   if freqs[word]:
      for emotion, val in emos.iteritems():
         distinctive_emotion_words.append([emotion, word, val,
             freqs[word]])
   else:
      for emotion, val in emos.iteritems():
         distinctive_emotion_words.append([emotion, word, val, 0])
#frequenza:
sort_for_frequency = sorted(distinctive_emotion_words, key=lambda
    element: (element[3]), reverse = False)
#distintivita:
sort_for_value = sorted(sort_for_frequency, key=lambda element:
    (element[2]), reverse = True)
#emozione
sort_for_emotion = sorted(sort_for_value, key=lambda
    element:(element[0]))
```

```
sorted_distinctive_all = sort_for_emotion
sorted_distinctive_threshold = []
freq_threshold = 5
for elem in sorted_distinctive_all:
   if int(elem[3]) >= freq_threshold:
   sorted_distinctive_threshold.append(elem)
emotion_words_dictionary = defaultdict(list)
for elem in emo_list_tot:
   emotion = elem[0]
   word = elem[1]
   value = elem[2]
   emotion_words_dictionary[emotion].append([word,
       Decimal(value)])
for emotion in sorted(emotions):
   sorted_for_emotion = sorted(emotion_words_dictionary[emotion],
       key=lambda element: (element[1]), reverse = True)
   sorted_for_emotion_word = []
   for elem in sorted_for_emotion:
      sorted_for_emotion_word.append(elem[0])
   emotion_words_dictionary[emotion] = sorted_for_emotion_word


def percentage_split(seq, percentages):
   cdf = cumsum(percentages)
   assert cdf[-1] == 1.0
   stops = map(int, cdf * len(seq))
   return [seq[a:b] for a, b in zip([0]+stops, stops)]


#creo un dizionario che contiene il nome del quartile e tutte le
    parole che ne fanno parte:
#{quartile_q4_emozione: [w1, w2,...], quartile_q3_emozione[w1,
    w2]}
quartiles = defaultdict(list)
```

```
for emotion in emotion_words_dictionary:
   quart =
       list(percentage_split(emotion_words_dictionary[emotion],
       [0.25]*4))
   j = 5
   for i in range(1,5):
      keyname = str(emotion)+"_q"+str(j-i)
      quartiles[keyname] = quart[i-1]


def polarity(lemmaPos):
   polarity_score =
       ((Decimal(emolex_all[lemmaPos]["gioia"])+Decimal(emolex_all[lemmaPos]["fi
       -
       ((Decimal(emolex_all[lemmaPos]["disgusto"])+Decimal(emolex_all[lemmaPos][
       Decimal(emolex_all[lemmaPos]["rabbia"])+Decimal(emolex_all[lemmaPos]["tri
   return polarity_score


def Freq_Seeds_Creation():
   #variabile per determinare quanti seed per ogni emozione
   seed_number = 4
   top_seeds = defaultdict(lambda: defaultdict(int))
   for emotion in sorted(emotions):
      top_seed =[]
      lista_tot = []
         for elem in sorted_emo_seeds_freq:
            if elem[1] == emotion:
               lista_tot.append(elem)
      top_seed.append(lista_tot[0:seed_number])
      for emozione in top_seed:
         for tripla in emozione:
            wordpos = unicode(tripla[0]+"-"+tripla[2]).lower()
            top_seeds[emotion][wordpos] = 0
```

```
   return top_seeds


#funzione che crea un dizionario con le top parole piu distintive
    per ogni emozione
def Dist_words_Creation(sorted_distinctive):
   dist_treshold = 0
   distinctive_words_per_emotion = 4
   top_distinctive_words = defaultdict(lambda: defaultdict(int))
   for emotion in sorted(emotions):
      top_distinctive = []
      lista_tot = []
      for elem in sorted_distinctive:
         if elem[0] == emotion:
            lista_tot.append(elem)
      top_distinctive.append(lista_tot[0:distinctive_words_per_emotion])
      for emozione in top_distinctive:
         for tripla in emozione:
            top_distinctive_words[emotion][tripla[1]]
   return top_distinctive_words

def quartiles_count_create():
   quartiles_count = defaultdict(int)
   for keyname in quartiles:
      quartiles_count[keyname] = 0
   return quartiles_count

n_top_emotive_words = 4
def top_emo_words():
   top_emos = defaultdict(lambda: defaultdict(int))
   for i in range(0, n_top_emotive_words):
      for emotion in sorted(emotions):
         top_emos["W"+str(i)][emotion] = 0
```

```
   return top_emos

emotivity_threshold = 0.4
def threshold_check(word, emolex, threshold):
   check = False
   for emotion in emotions:
      if (float(emolex[word][emotion]) - threshold) >0:
         check = True
   return check

with codecs.open("evalitaCleanTweets.pos", "r", encoding="utf-8")
    as corpus, codecs.open("extracted_features_new_2.csv", "w+",
    encoding="utf-8") as outfile:
   first_Pass = True
   for i, line in enumerate(corpus):
      #se la linea inzia col doc so che e un nuovo tweet, azzero i
          valori che mi servono
      if line.startswith("<doc"):
         output = []
         output_dict = defaultdict(str)
         quartiles_count = quartiles_count_create()
         splitted_line = line.split('"')
         tweet_id = splitted_line[1]
         #numero di parole
         words = 0
         #numero di parole emotive
         emo_words = 0
         #numero di parole emotive filtrate
         emo_words_thresh = 0
         #numero di parole piene
         content_words = 0
```

```
        #dizionari per la presenza di parole nei seed o nelle
            parole piu distintive per ogni emozione
        top_words_all =
            Dist_words_Creation(sorted_distinctive_threshold)
        top_seeds = Freq_Seeds_Creation()
        #dizionario per salvare il valore massimo di ogni emozione
        emotions_max = defaultdict(int)
        for emotion in emotions:
           emotions_max[emotion] = 0
        #numero di parole positive e negative
        positive_words = 0
        negative_words = 0
        emotive_words = []
        top_w_emo = top_emo_words()
        polarities = []
    #se la linea rappresenta la fine di un tweet scrivo i valori
        sul file di output
    elif line.startswith("</doc"):
        output.append(["Twitter_id", str(tweet_id)])
        output.append(["COUNT_emoWords_no_threshold",
            str(emo_words)])
        output.append(["COUNT_emoWords_on_Tot",
            str(emo_words/float(words))])
        output.append(["COUNT_emoWords_Threshold",
            str(emo_words_thresh)])
        if content_words > 0:
           output.append(["SCORE_emotivity_score",
               str(emo_words_thresh/float(content_words))])
        else:
           output.append(["SCORE_emotivity_score", str(0)])
        for emotion in sorted(emotions):
```

```
   output.append(["MAX_"+emotion+"",
       str(emotions_max[emotion])])
for quartile in sorted(quartiles_count):
   quartile_name = quartile
   splitted_quartile = quartile_name.split("_")
   emotion,quart = splitted_quartile
   output.append(["QUARTILE_"+quart+"_"+emotion,str(quartiles_count[qua
for emotion, values in top_seeds.iteritems():
   for lemmapos, value in values.iteritems():
      output.append(["SEED_"+emotion+"_"+lemmapos,
          str(value)])
for emotion, values in top_words_all.iteritems():
   for lemmapos, value in values.iteritems():
      output.append(["DISTINCTIVE_ALL_"+emotion+"_"+lemmapos,
          str(value)])
output.append(["COUNT_positive_words",
    str(positive_words)])
output.append(["COUNT_negative_words",
    str(negative_words)])
while len(polarities) < 20:
   polarities.append(0)
for i, elem in enumerate(polarities):
   output.append(["POLARITIES_W"+str(i+1)+"", str(elem)])
top_emotive_words = sorted(emotive_words, key=lambda x:
    x[2], reverse = True)[0:n_top_emotive_words]
for i in range(0, n_top_emotive_words):
   w_number = i
   for emotion, value in
       top_w_emo["W"+str(w_number)].iteritems():
      try:
         if top_emotive_words[i][1] == emotion: #il
             problema e qui
```

```
                top_w_emo["W"+str(w_number)][emotion] = 1
            except:
                pass
    for w, values in top_w_emo.iteritems():
        for emotion, value in values.iteritems():
            output.append(["TOP_"+w+"_"+emotion+"",
                top_w_emo[w][emotion]])
    if first_Pass:
        for elem in output:
            outfile.write(unicode(elem[0]))
            outfile.write("\t")
        outfile.write("\n")
        for elem in output:
            outfile.write(unicode(elem[1]))
            outfile.write("\t")
        outfile.write("\n")
        first_Pass = False
    else:
        for elem in output:
            outfile.write(unicode(elem[1]))
            outfile.write("\t")
        outfile.write("\n")
#se la linea non e vuota estraggo le feature
elif line.strip():
    splitted = line.split("\t")
    unicode(splitted)
    idx,word,lemma,pos,pos_fine,other = splitted
    lemmapos = lemma.lower()+"-"+pos.lower()
    words += 1
    #se presente in parole emotive aggiungo al conto delle
        parole emotive
    if lemmapos in emolex_all:
```

```
emo_words+=1
#se ha soglia sufficiente di emotivita aggiungo al
    conto
if threshold_check(lemmapos, emolex_all,
    emotivity_threshold):
   emo_words_thresh+=1
#se parola piena aggiungo al conto
if pos in content_POS:
   content_words+=1
#se una delle sue emozioni e il massimo trovato
    aggiorno il valore
for emotion in emotions_max:
   if
       float(emolex_all[lemmapos][emotion])-float(emotions_max[emotio
       > 0:
      emotions_max[emotion] =
          emolex_all[lemmapos][emotion]
#aggiorno il valore della presenza nei quartili delle
    emozioni
for key in quartiles_count:
   if lemmapos in quartiles[key]:
      quartiles_count[key] += 1
#controllo la polarita della parola e aggiungo al conto
polarity = polarity(lemmapos)
   if polarity > 0:
      positive_words+=1
else:
   negative_words+=1
polarities.append(polarita)
#aggiungo la parola e la sua emozione dominante alla
    lista totale per avere poi i top X
```

```
            max_emo = max(emodist_all[lemmapos].iteritems(),
                key=operator.itemgetter(1))[0]
            emotive_words.append([lemmapos, max_emo,
                emodist_all[lemmapos][max_emo]])
            #controllo se e presente nei top seeds
            for emotion, values in top_seeds.iteritems():
               for key, value in values.iteritems():
                  if   lemmapos == key:
                  #deve controllare se uguale, non se presente e
                      bastsa
                     top_seeds[emotion][key]+=1
            #controllo se e presente nelle parole piu rappr totali
                e per lemma
            for emotion, values in top_words_all.iteritems():
               for key, value in values.iteritems():
                  if   lemmapos == key:
                     top_words_all[emotion][key]+=1
         #stampa ogni 1000 caratteri per avere a video un'idea
             dello stato di esecuzione
         if i%1000 == 0:
         print i
```

Appendice C

Risultati Training Set

C.1 Risultati modelli standard

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.73	0.68	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.1: Risultati del training-set sul modello Full

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.35	0.45
NEG	0	0.74	0.69	0.71
	1	0.63	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.2: Risultati del training-set sul modello NoDistinctive

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.34	0.44
NEG	0	0.73	0.68	0.7
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.7

Tabella C.3: Risultati del training-set sul modello NoSeed

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.36	0.45
NEG	0	0.73	0.68	0.71
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.4: Risultati del training-set sul modello NoScore

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.76	0.95	0.84
	1	0.64	0.23	0.34
NEG	0	0.73	0.58	0.64
	1	0.57	0.72	0.63
GLOBAL		0.69	0.69	0.67

Tabella C.5: Risultati del training-set sul modello Solo Conteggi (Count, Score, PosCount, NegCount)

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.35	0.44
NEG	0	0.74	0.67	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.6: Risultati del training-set sul modello EmoDistribution(Max, Quartile, Top)

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.35	0.44
NEG	0	0.74	0.68	0.71
	1	0.7	0.66	0.71
GLOBAL		0.71	0.72	0.71

Tabella C.7: Risultati del training-set sul modello NoSeedNoDistinctive

C.2 Risultati modelli con spazio normalizzato per parola

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.37	0.46
NEG	0	0.74	0.68	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.8: Risultati del training-set sul modello Full

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.37	0.46
NEG	0	0.74	0.68	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.9: Risultati del training-set sul modello NoDistinctive

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.44
NEG	0	0.74	0.68	0.71
	1	0.62	0.7	0.66
GLOBAL		0.71	0.72	0.71

Tabella C.10: Risultati del training-set sul modello NoSeed

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.36	0.45
NEG	0	0.74	0.68	0.71
	1	0.63	0.69	0.66
GLOBAL		0.71	0.72	0.71

Tabella C.11: Risultati del training-set sul modello NoScore

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.76	0.95	0.84
	1	0.64	0.23	0.34
NEG	0	0.73	0.58	0.64
	1	0.57	0.72	0.63
GLOBAL		0.69	0.69	0.67

Tabella C.12: Risultati del training-set sul modello Solo Conteggi (Count, Score, PosCount, NegCount)

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.35	0.44
NEG	0	0.74	0.67	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.13: Risultati del training-set sul modello EmoDistribution(Max, Quartile, Top)

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.61	0.35	0.44
NEG	0	0.74	0.68	0.71
	1	0.62	0.7	0.66
GLOBAL		0.71	0.72	0.71

Tabella C.14: Risultati del training-set sul modello NoSeedNoDistinctive

C.3 Risultati modelli con spazio normalizzato Z-score

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.74	0.68	0.71
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.15: Risultati del training-set sul modello Full

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.35	0.45
NEG	0	0.74	0.68	0.71
	1	0.62	0.68	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.16: Risultati del training-set sul modello NoDistinctive

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.34	0.44
NEG	0	0.74	0.68	0.71
	1	0.62	0.69	0.65
GLOBAL		0.71	0.72	0.7

Tabella C.17: Risultati del training-set sul modello NoSeed

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.91	0.84
	1	0.62	0.36	0.46
NEG	0	0.74	0.69	0.71
	1	0.63	0.69	0.65
GLOBAL		0.71	0.72	0.71

Tabella C.18: Risultati del training-set sul modello NoScore

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.76	0.95	0.84
	1	0.65	0.24	0.34
NEG	0	0.72	0.58	0.64
	1	0.57	0.72	0.63
GLOBAL		0.69	0.69	0.67

Tabella C.19: Risultati del training-set sul modello Solo Conteggi (Count, Score, PosCount, NegCount)

Task	*Class*	*Precision*	*Recall*	*F-score*
POS	0	0.78	0.92	0.84
	1	0.62	0.34	0.44
NEG	0	0.73	0.67	0.7
	1	0.62	0.68	0.65
GLOBAL		0.71	0.71	0.7

Tabella C.20: Risultati del training-set sul modello EmoDistribution(Max, Quartile, Top)

Bibliografia

[1] J. C. D. Albornoz, L. Plaza, and P. Gervas. Improving emotional intensity classification using word sense disambiguation. *Natural Language Processing and its Applications. Research in Computing Science*, (46):131–142, 2010.

[2] F. Auerbach. Das gesetz der bevolkerungskonzentration. *Petermanns Geographische Mitteilungen*, 59:74–76, 1913.

[3] S. Baccianella, A. Esuli, and F. Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *in Proc. of LREC*, 2010.

[4] L. F. Barrett. Solving the emotion paradox: Categorization and the experience of emotion. *Personality and Social Psychology Review*, (10):20–46, 2006.

[5] V. Basile, A. Bolioli, and M. Nissim. Overview of the evalita 2014 sentiment polarity classification task. In *Proceedings of the First Italian Conference on Computational Linguistics CLiC-it 2014 and the Fourth International Workshop EVALITA 2014*, pages 50–57, Pisa, dec 2014. Pisa University Press.

[6] V. Basile and M. Nissim. Sentiment analysis on italian tweets. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 100–107, 2013.

[7] E. Bicici and D. Yuret. Clustering word pairs to answer analogy questions. In *In Proceedings of the Fifteenth Turkish Symposium on Artificial Intelligence and Neural Networks (TAINN 2006)*, pages 143–152, Akyaka, Mugla, Turkey, 2006.

[8] H. Binali and V. Potdar. Emotion detection state of the art. In *CUBE '12: Proceedings of the CUBE International Information Technology Conference*, pages 501–507, 2012.

[9] H. Binali, C. Wu, and V. Potdar. Computational approaches for emotion detection in text. In *Proceedings of IEEE DEST 2010*, page 6, Dubai, 2010.

[10] C. Bosco, V. Patti, and A. Bolioli. Developing corpora for sentiment analysis: The case of irony and senti-tut. *IEEE Intelligent Systems, Special Issue on Knowledge-based Approaches to Contentlevel Sentiment Analysis*, 28(2):55–63, 2013.

[11] R. Burget, J. Karasek, and Z. Smekal. Recognition of emotions in czech newspaper headlines. *Radioengineering*, 20:39–47, apr 2011.

[12] M. J. Cafarella, M. Banko, and O. Etzioni. Relational web search. Tech. rep., University of Washington, Department of Computer Science and Engineering, 2006. 2006-04-02.

[13] Pew Research Center. News use across social media platforms, nov 2013.

[14] P. A. Chew, B. W. Bader, T. G. Kolda, and A. Abdelali. Cross-language information retrieval using parafac2. In ACM Press, editor, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD07)*, pages 143–152, Borovets, Bulgaria, 2007.

[15] T. Chklovski and P. Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of Experimental Methods in Natural Language Process- ing 2004 (EMNLP-04)*, pages 33–40, Barcelona, Spain, 2004.

[16] K. Church and P. Hanks. Word association norms, mutual information, and lexicography. In *In Proceedings of the 27th Annual Conference of the Association of Computational Linguistics*, pages 76–83, Vancouver, British Columbia, 1989.

[17] H. I. Chyi and D. L. Lasorsa. An explorative study on the market relation between online and print newspapers. *Journal of Media Economics*, 15(2):91–106, 2002.

[18] D. Das and S. Bandyopadhyay. Sentence level emotion tagging. In *Affective Computing and Intelligent Interaction and Workshops*, pages 1–6. ACII 2009, 2009.

[19] S. C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science (JASIS)*, 41(6):391–407, 1990.

[20] C. Fellbaum (Ed.). *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[21] P. Ekman. *An Argument for Basic Emotions. Cognition and Emotion.* 1992.

[22] Inc. Facebook. Graph api overview, 2016.

[23] H. Gunes and M. Pantic. Automatic, dimensional and continuous emotion recognition. *International journal of synthetic emotions*, 1:68, 2010.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[25] Z. S. Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

[26] Z. S. Harris. *Mathematical structures of language*. Wiley, New Jersey, 1968.

[27] N. Janmenjoy, N. Bighnaraj, and H. S. Behera. A comprehensive survey on support vector machine in data mining tasks: Applications & challenges. *International Journal of Database Theory and Application*, 8(1):169–186, 2015.

[28] X. Jin and Z. Wang. In T. Tan J. Tao and R. Picard, editors, *An Emotion Space Model for Recognition of Emotions in Spoken Chinese Affective Computing and Intelligent Interaction*, volume 3784, pages 397–402. Springer Berlin / Heidelberg, 2005.

[29] T. Joachims. *Advances in Kernel Methods - Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. B. Scho?lkopf, C. Burges, and A. Smola, 1999.

[30] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[31] F. Keshtkar and D. Inkpen. A corpus-based method for extracting paraphrases of emotion terms. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 35–44, Los Angeles, California, 2010. Association for Computational Linguistics.

[32] G. Lakoff. *Women, Fire, and Dangerous Things.* University Of Chicago Press, Chicago, IL, 1987.

[33] T. K. Landauer and S. T. Dumais. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

[34] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774. Association for Computational Linguistics, 1998.

[35] D. Lin and P. Pantel. Dirt ? discovery of inference rules from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining 2001*, pages 323–328, 2001.

[36] B. Liu. *Handbook of Natural Language Processing, Second Edition*, chapter Sentiment Analysis and Subjectivity. N. Indurkhya and F.J. Damerau, 2010.

[37] B. Liu. *Sentiment Analysis and Opinion Mining.* Morgan & Claypool Publishers, may 2012.

[38] W. Lowe. Towards a theory of semantic space. In *Proceedings of the Twenty-first Annual Conference of the Cognitive Science Society*, pages 576–581, 2001.

[39] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*, 28(2):203–208, 1996.

[40] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[41] S. M. Mohammad and P.D. Turney. Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, 29(3):436–465, 2013.

[42] T. Nasukawa and J. Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the K- CAP-03, 2nd Intl. Conf. on Knowledge Capture*, 2003.

[43] A. Neviarouskaya, H. Prendinger, and M. Ishizuka. Recognition of fine-grained emotions from text: An approach based on the compositionality principle. In e C. Faucher T. Nishida, L. Jain, editor, *Modelling Machine Emotions for Realizing Intelligence: Foundations and Applications*, volume 1, pages 179–207. Springer Smart Innovation, Systems and Technologies (SIST), 2010.

[44] N. Newman, W. H. Dutton, and G. Blank. Social media in the changing ecology of news: The fourth and fifth estates in britain. *International Journal of Internet Science*, 7(1):6–22, 2012.

[45] Y. Niwa and Y. Nitta. Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In *Proceedings of the 15th International Conference On Computational Linguistics*, pages 304–309, Kyoto, Japan, 1994.

[46] S. PadÚ and M. Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

[47] B. Pang and L. Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[48] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2002)*, 2002.

[49] P. Pantel and D. Lin. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619, Edmonton, Canada, 2002a.

[50] P. Pantel and D. Lin. Document clustering with committees. In *Proceedings of the 25th Annual International ACM SIGIR Conference*, pages 199–206, 2002b.

[51] P. Pantel and P.D. Turney. From frequency to meaning: vector space models for semantics. *Journal of Artificial Intelligence Research*, 37:141–188, 2010.

[52] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Names and similarities on the web: Fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 809–816, Sydney, Australie, 2006.

[53] L. C. Passaro, G. E. Lebani, L. Pollacci, Emmanuele Chersoni, and A. Lenci. The coling lab system for sentiment polarity classification of tweets. In *Proceedings of the Fourth International Workshop EVALITA 2014*, pages 87–92, dec 2014.

[54] L. C. Passaro, L. Pollacci, and A. Lenci. Item: A vector space model to bootstrap an italian emotive lexicon. In *Proceedings of the second Italian Conference on Computational Linguistics CLiC-it 2015*, pages 215–220, Trento - Italy, 2015.

[55] T. Pedersen. *Word Sense Disambiguation: Algorithms and Applications*, chapter Unsupervised corpus-based methods for WSD. Springer, 2006.

[56] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, pages 183–190, Honolulu, HI, 1993.

[57] R. Picard. Blogs, tweets, social media, and the news business. *Nieman Reports*, 63(3):10–12, 2009.

[58] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf e C. Burges e A. Smola, editor, *Advances in Kernel Methods*, pages 185–208. 1998.

[59] R. Plutchik. *The psychology and biology of emotion*. Harper Collins, New York, 1994.

[60] C. Quan and F. Ren. Sentence emotion analysis and recognition based on emotion words using rencecps. *International Journal of Advanced Intelligence*, 2:105–117, jul 2010.

[61] R. Rapp. Word sense discovery based on sense descriptor dissimilarity. In *Proceedings of the Ninth Machine Translation Summit*, pages 315–322, 2003.

[62] L. Renzi, G. Salvi, and A. Cardinaletti. *Grande grammatica italiana di consultazione*. Il Mulino, Bologna, 2001.

[63] E. Rosch and B. Lloyd. *Cognition and Categorization*. Lawrence Erlbaum, Hillsdale, NJ, 1978.

[64] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[65] D. V. Shah, J. Cho, W. P. Eveland, and N. Kwak. Information and expression in a digital age. *Communication Research*, 32(10):531–565, 2005.

[66] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[67] J. Shawe-Taylor and N. Cristianini. *Support Vector Machines*. Cambridge University Press, 2000.

[68] N. M. Shelke. Approaches of emotion detection from text. *International Journal of Computer Science and Information Technology Research*, 2(2):123–128, jun 2014.

[69] S. N. Sivanandam, S. Sumathi, and S.N. Deepa. *Introduction to Neural Networks using MATLAB 6.0*. Tata McGraw Hill Education Pvt. Ltd., 2006.

[70] R. Snow, D. Jurafsky, and A. Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 801–808, 2006.

[71] C. Strapparava and R. Mihalcea. Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560, Fortaleza, Brazil, 2008. Association for Computational Linguistics, ACM, 2008.

[72] P. N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[73] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-01)*, pages 491–502, Freiburg, Germany, 2001.

[74] P. D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, 2002.

[75] P. D. Turney and M. L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 21(4):315–346, 2003.

[76] P. D. Turney, M. L. Littmann, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems.

In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, pages 482–489, Borovets, Bulgaria, 2003.

[77] R. Valitutti. Wordnet-affect: an affective extension of wordnet. In *In Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086, 2004.

[78] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[79] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons.

[80] E. Vozalis and K. Margaritis. Analysis of recommender systems algorithms. In *Proceedings of the 6th Hellenic European Conference on Computer Mathematics and its Applications (HERCMA-2003)*, Athens, Greece, 2003.

[81] w. Zhang, T. Yoshida, and X. Tang. Text classification based on multi-word with support vector machine. *Knowledge-Based Systems*, 21:879–886, 2008.

[82] C. H. Wan, L. H. Lee, R. Rajkumar, and D. Isa. A hybrid text classification approach with low dependency on parameter by integrating k-nearest neighbor and support vector machine. *Expert Systems with Applications*, 39(15):11880?–11888, 2012.

[83] J. R. Wiebe, F. Bruce, and T. P. O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the Association for Computational Linguistics (ACL-1999)*, 1999.

[84] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, 2011.

[85] C. Yang, K.H.Y. Lin, and H.H. Chen. Emotion classification using web blog corpora. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2007.

[86] M. Yassine and H. Hajj. A framework for emotion mining from text in online social networks. In *Data Mining Workshops (ICDMW)*, pages 1136–1142. 2010 IEEE International Conference, 2010.

[87] E. A. Zanaty. Support vector machines (svms) versus multilayer perception (mlp) in data classification. *Egyptian Informatics Journal*, 13:177–183, 2012.

[88] M. Zhitomirsky-Geffet and I. Dagan. Bootstrapping distributional feature vector quality. *Computational Linguistics*, 35(3):435–461, 2009.

[89] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Reading, MA, 1949.