Alma Mater Studiorum - Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

ANALISI DELLA POLARITÀ SU CHAT DI LIVESTREAMING

Relatore:
Chiar.mo Prof.
Fabio Tamburini

Presentata da:
Federico Foschini

III Sessione
Anno Accademico 2014/2015

A Francesco e Silvia

Introduzione

L'obiettivo del presente lavoro di tesi è stato quello di progettare un sistema di *sentiment analysis* in grado di rilevare e classificare le opinioni e i sentimenti espressi tramite chat dagli utenti della piattaforma di video streaming Twitch.tv. Per impostare ed organizzare il lavoro, giungendo quindi alla definizione del sistema che ci si è proposti di realizzare, sono stati utilizzati vari modelli di analisi, applicandoli, dapprima, su dati etichettati in maniera automatica attraverso l'uso di emoticon e, successivamente, su dati etichettati a mano: l'elaborato, pertanto, espone i risultati scaturiti da tale studio. Verrà infine presentato il sistema utilizzato, illustrando come lo stesso sia in grado di valutare l'andamento giornaliero di uno specifico *video streaming*, attraverso l'analisi dei commenti espressi dagli spettatori.

Sentiment Analysis

Con il termine *Sentiment analysis*, o *opinion minig*, si intende uno studio che analizza le opinioni, i sentimenti, le valutazioni e le emozioni del linguaggio utilizzato dalle persone. Spesso il concetto di *sentiment analysis* si estende anche all'ambito del *data mining* e del *text mining*, ossia quella branca di ricerca riguardante l'estrazione di un sapere o di una conoscenza a partire da grandi quantità di dati.

La *Sentiment analysis*, nata nell'ambito dell'informatica, è una delle aree di ricerca più attive nel campo dell'analisi del linguaggio naturale e si è diffusa ampiamente anche in altri rami scientifici come ad esempio la scienza

sociale, l'economia e il marketing. Inoltre questo tipo di ricerca ha avuto un grande impatto sia a livello commerciale, per aziende e multinazionali, che sociologico, coinvolgendo l'intera società che ci circonda.

L'enorme diffusione della *sentiment analysis* coincide con la crescita dei cosiddetti *social media*: siti di commercio e recensioni di prodotti, forum di discussione, blog, micro-blog e di vari *social network*. Per la prima volta nella storia è possibile disporre di una enorme quantità di dati digitalizzati. Dati da manipolare, analizzare e studiare in maniera estremamente approfondita, al fine di estrapolare ed ottenere un'analisi ampiamente sfaccettata e particolareggiata.

Gli studi di *sentiment analysis* sono applicabili in quasi tutte le attività del cosiddetto dominio sociale, le opinioni e i pareri sono infatti al centro della maggior parte delle attività umane e sono altresì la chiave che influenza il nostro comportamento. Ciò in cui crediamo, la percezione della realtà, le scelte che facciamo sono condizionate dall'ambiente che ci circonda e dai giudizi degli altri riguardo noi stessi. Per questo motivo, quando abbiamo bisogno di prendere una decisione, spesso chiediamo il parere e il consenso altrui. Questo comportamento si verifica non solo nei singoli individui, ma lo possiamo vedere applicato allo stesso modo ad organizzazioni, aziende e grandi multinazionali. Da tutti questi aspetti è facile dedurre come, *la sentiment analysis* possa avere una grande influenza nel mondo che ci circonda sia dal punto di vista economico che sociale.

Mentre la linguistica e l'analisi del linguaggio naturale (NLP) hanno una lunga storia, la *sentiment analysis* è relativamente recente: le prime ricerche sono state svolte a partire dall'anno 2000. Nonostante questo ambito di ricerca sia così giovane si presenta come un'area estremamente attiva. La crescita della ricerca nel campo della *sentiment analysis*, avvenuta in maniera così immediata e su larga scala, ha molteplici ragioni. Per prima cosa, le applicazioni pratiche di questo tipo di ricerca sono svariate e possono essere utilizzate in un ampio raggio di situazioni e domini differenti. Ciò ha generato un aumento di interesse notevole da parte di aziende e di grandi

multinazionali (non solo in ambito informatico) con un relativo stanziamento di finanziamenti specifici per questo campo di ricerca. In secondo luogo, il grande interesse dimostrato da parte del mercato ha portato ad una proliferazione di applicazioni commerciali e ad una costante ricerca della soluzione di problemi che fino ad ora non erano ancora stati studiati.

Applicazioni della Sentiment Analysis

Partiamo dunque dal presupposto che le opinioni e i pareri siano la base di tutte le attività sociali, essendo la chiave che influenza il nostro comportamento. Nel mondo reale, aziende e multinazionali sono costantemente alla ricerca di giudizi e pareri dati dai consumatori rispetto ai loro prodotti. Allo stesso modo il consumatore vuole avere informazioni che riguardino gli articoli da acquistare, opinioni inerenti un determinato film o un nuovo album musicale e così via. In passato, quando un'azienda necessitava di un parere del consumatore, effettuava sondaggi d'opinione o gruppi di discussione: ottenere questo tipo di dati è stato per lungo tempo un grande business per aziende di marketing e per compagnie pubblicitarie. Al giorno d'oggi chiunque voglia comprare un prodotto, potendo trovare centinaia di recensioni o discussioni online non è più limitato a dover chiedere informazioni ad amici o famigliari. Allo stesso modo le aziende non sono più costrette a condurre sondaggi o analisi di mercato vista l'enorme diffusione dei dati disponibili. Tuttavia la ricerca, l'estrazione e l'annotazione di questi dati è un compito di una complessità non indifferente, basti pensare alla quantità di informazioni scritte nel post di un blog o in un forum, che rendono difficoltosa anche ad un singolo l'elaborazione e la sintesi delle informazioni utili: ecco perché diventa cosi importante uno studio finalizzato e strutturato in un determinato ambito.

La piattaforma web Twitch.tv

La piattaforma web Twitch.tv [1] è nata nel 2011 come *spin off* del sito di *video streaming* Justin.tv ed è stata attualmente comprata da Amazon.com. Gli ambiti specifici a cui si dedica sono video prodotti in maniera amatoriale dagli utenti stessi e possono avere come oggetto i videogiochi e le attività ludiche; più recentemente sono stati introdotti anche argomenti di carattere maggiormente creativo, come la pittura, la musica o il modellismo.

Dal mese di Ottobre 2013 ad oggi, la popolarità di questa piattaforma è cresciuta molto rapidamente, fino ad arrivare ad eclissare completamente il sito stesso da cui derivava, Justin.tv, che, per questo motivo, è stato dismesso al fine del 2014. Per quantificare in maniera oggettiva la portata di questa crescita, basti pensare che, alla fine del 2015, Twitch.tv ha registrato più di un milione e mezzo di trasmissioni in contemporanea visualizzate da più di cento milioni di visitatori mensili: questa straordinaria crescita ha fatto sì che il sito, a livello mondiale, si collocasse al quarto posto nella classifica che valuta il traffico internet generato (Wall Street Journal 2015).

Esaminando il livello di funzionalità del sito, si può osservare come lo stesso presenti una *homepage* composta da diversi sezioni, ognuna delle quali rappresenta un determinato videogioco o particolare attività, come, ad esempio, la pittura. Entrando nella sezione scelta si possono visualizzare tutti i canali attivi, che non sono altro che una trasmissione in tempo reale da parte di un utente di Twitch: questo utente assume il ruolo di "conduttore " o, per utilizzare il termine specifico della piattaforma stessa, di *broadcaster*. In genere il *broadcaster* trasmette i contenuti riprodotti nel proprio computer, mentre, nel caso di attività pittoriche o musicali, si ricorre alla registrazione di video attraverso l'utilizzo di una videocamera. Ciò consente agli spettatori di seguire e commentare gli avvenimenti in tempo reale, grazie ad una chat testuale che permette loro di interagire direttamente con il conduttore: nella presente tesi sono stati raccolti proprio questi dati, che sono stati poi

[1] `http://wwww.twitch.tv`

utilizzati per effettuare valutazioni e analisi sulla polarità dei contenuti.

Organizzazione del lavoro

Si ritiene importante, a questo punto, presentare una sintesi dei contenuti trattati in questa tesi:

- Capitolo 1: in questo capitolo viene introdotto il tema della *sentiment analysis* e viene definito formalmente il problema della *sentiment classification*. Inizialmente saranno presentati i vari metodi di approccio al problema e le relative ricerche che illustrano i risultati ottenuti; nella seconda parte del capitolo saranno analizzati nello specifico i modelli e gli strumenti utilizzati nella tesi.

- Capitolo 2: il secondo capitolo presenta il progetto della tesi, partendo dalla descrizione della metodologia applicata, per giungere alla normalizzazione del testo e mostare, infine, i risultati ottenuti attraverso l'utilizzo dei modelli descritti. Nella parte finale verrà effettuata un'analisi sul trend giornaliero delle trasmissioni di alcuni utenti, focalizzando l'attenzione sull'andamento della polarità nella chat.

- Capitolo 3: quest'ultimo capitolo, contenente le conclusioni della tesi, illustra gli aspetti positivi e negativi del sistema utilizzato e delinea i miglioramenti che potrebbero essere apportati e i possibili, futuri sviluppi.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Approcci alla sentiment analysis

Lo scopo di questo capitolo è quello di introdurre i concetti base della *sentiment analysis*, presentare la varie metodologie utilizzate in questo tipo di analisi e descriverle brevemente per poi analizzarle, soffermandoci in particolare su quelle in cui sono presenti gli stessi strumenti utilizzati in questa tesi: le reti neurali ricorsive (*RNNLM*), la rappresentazione vettoriale di parole *word embedding* con (*word2vec*) e i classificatori lineari (*Liblinear*).

1.1 Definizione del problema

In questo capitolo verranno introdotti i concetti base e le astrazioni necessarie per descrivere l'argomento trattato. È bene partire dal presupposto che, rispetto alle informazioni oggettive, le opinioni e l'espressione dei sentimenti sono soggettive: per questo motivo è molto importante raccogliere i pareri e le opinioni di più persone in modo da ottenere un risultato il più possibile oggettivo.

Consideriamo, ad esempio, la seguente recensione di un prodotto:

> (1) Ho comprato un computer sei mesi fa. (2) Funziona molto bene. (3) La potenza di calcolo è soddisfacente. (4) Il prezzo è

accettabile. (5) Purtroppo è troppo pesante.

Analizzandola, possiamo notare che ci troviamo di fronte ad una serie di opinioni: la frase (2) esprime un parere generale sul prodotto, le frasi (3) e (4) esprimono opinioni positive riguardo a particolari aspetti del prodotto stesso, la frase (5) esprime una opinione negativa.

Da questa prima analisi risulta evidente come, in una serie di frasi che esprimono un'opinione possiamo trovare due diverse componenti: il soggetto su cui verte l'opinione g e l'opinione espressa s. Questa tipologia di frase, può essere rappresentata come una coppia di questi due valori (g, s). Il valore s può esprimere sia parere positivo, negativo o neutro (definito *polarity*) oppure un valore che esprime l'intensità di questo sentimento (Es: nel caso delle recensioni potrebbe essere un valore espresso tra 1 e 5 stelle)

Definizione 1.1.1. (opinione): un'opinione è una quadrupla (g, s, h, t) dove g rappresenta un parere (o sentimento) riguardo ad particolare oggetto o persona, s è il sentimento espresso, h è l'entità che esprime questa opinione e t rappresenta il segmento di tempo in cui l'opinione stessa viene espressa.

Anche se questa definizione è rappresentata in modo molto conciso non è facile applicarla a casi reali: tornando alla recensione sopra citata si può notare, ad esempio, che la frase (4) comunica che il prezzo del computer acquistato viene ritenuto accettabile. Questa opinione non ha come soggetto il prezzo, ma si riferisce al computer citato nella frase (1). Tenendo a mente questa considerazione notiamo come sia possibile scomporre il *target* della recensione sotto vari aspetti: possiamo, ad esempio definire la coppia (computer, prezzo) e, come proposto da Minqing and Liu (2004), assegnare una definizione formale a questo concetto:

Definizione 1.1.2. (entita): una entità e è un prodotto, servizio, argomento, persona o evento. Può essere descritto con una coppia (T, W), dove T rappresenta una gerarchia di parti, sottoparti e cosi' via e W rappresenta un insieme di attributi appratenenti ad e. Qualsiasi parte o sottoparte può avere un suo insieme distinto di attributi.

Considerando ancora una volta il nostro esempio, possiamo attribuire alla nostra entità *computer* una lista di attributi o caratteristiche: *velocità di calcolo, peso, costo*. Il nostro computer è anche composto, inoltre, da una serie di parti, per esempio, *lo schermo o la batteria*. Questa parti hanno a loro volta una serie di caratteristiche: la batteria, per esempio, possiede gli attributi per esprimere la propria *durata* e il proprio *peso*.

Si può facilmente intuire, pertanto, per quale motivo risulti cosi difficile effettuare questo tipo di analisi: è estremamente complicato, infatti, definire un'entità tenendo conto di tutte le parti e le caratteristiche che la contraddistinguono, a livelli differenti e in maniera dettagliata. Inoltre è importante sottolineare che molte applicazioni pratiche non hanno nemmeno bisogno di entrare cosi a fondo in tutta questa serie di dettagli. In genere, quindi, si utilizza un modello semplificato costituito ad albero, dove la radice dell'albero rappresenta l'entita stessa e il secondo livello (il livello delle foglie) rappresenta direttamente i differenti *aspect* dell'entità. Questo modello è quello che viene usato comunemente nella *sentiment analysis*.

Seguendo le ricerche presentate da Hu e Liu (2004) e successivamente da Liu (2010) possiamo definire cos'è un'opinione:

Definizione 1.1.3. (opinione): Un'opinione è una quitupla $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ dove e_i è il nome dell'entità, a_{ij} è una *aspect* di e_i, s_{ijkl} è il sentimento del'aspect a_{ij} dell'entità e_i, h_k è il soggetto che esprime l'opinione e t_l rappresenta il momento di tempo nel quale l'opinione viene espressa dal soggetto h_k. Come già detto precedentemente il sentimento s_{ijkl} può essere positivo, negativo o neutrale o esprimere una intesità, per esempio attraverso un valore da 1 a 5.

Nella definizione vengono esplicitate chiaramente le corrispondenze tra i vari elementi, e questo aspetto è particolarmente importante da sottolineare perché, qualora tali corrispondenze non vengano rispettate, la definizione perde di significato. Un altro aspetto rilevante da evidenziare consiste nel fatto che la mancanza di uno di questi elementi rende l'analisi delle opinioni molto meno specifica, in quanto rimuove una grossa fetta di informazioni. Nel

caso in cui, ad esempio, venga omesso il fattore tempo, si perderà il contesto temporale dell'opinione, benché sia diverso il valore tra una opinione espressa ieri rispetto ad una formulata due anni fa. È necessario, infine, specificare come la definizione data sia un modello semplificato del problema originale e se ciò rende la gestione di opinioni più semplice da trattare, comporta d'altro canto anche una perdita di informazioni. Se prendiamo, ad esempio, la seguente definizione: "Questa automobile è troppo piccola per una persona alta " noteremo che in essa non viene gestito il contesto dell'opinione: la circostanza che l'auto sia troppo piccola, infatti, è relativa soltanto alle persone alte e non dovrebbe quindi influenzare il giudizio generale.

Possiamo infine raggruppare i vari *task* di *sentiment analysis* nelle seguenti categorie:

- Categorizzazione ed estrazione delle entità: consiste nell'estrazione di tutte le entita D, e nel raggruppamento delle stesse per categorie o gruppi (*cluster*). Ogni *cluster* rappresenta un'entita e_i univoca.

- Categorizzazione ed estrazione degli *aspect*: riguarda l'estrazione di tutte le espressioni che rappresentano un *aspect* di una particolare entita raggruppandole per categorie o gruppi. Ognuno di questi gruppi rappresenta un *aspect* a_{ij} dell'entita e_i

- Categorizzazione ed estrazione del soggetto (*opinion holder*): si tratta dell'estrazione dei soggetti che esprimono le opinioni, e del loro raggruppamento per categorie. Questo *task* è analogo a quelli appena descritti.

- Estrazione e standardizzazione del tempo: consiste nell'estrazione dei momenti temporali durate i quali sono state espresse le opinioni e nell'effettuazione delle opportune conversioni per portare il tempo ad un formato standard. Anche questo *task* è analogo a quelli visti finora.

- Classificazione degli *aspect*: consiste nella determinazione di una opinione o *aspect* a_{ij} in base ad una valutazione positiva, negativa, neutrale o nell'assegnazione di un valore numerico *rating*.
- Generazione delle quintuple che rappresentano le opinioni: riguarda la costruzione delle quintuple $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ definite in precedenza.

1.2 Document Sentiment Classification

Dopo un approccio di tipo *aspect based*, passiamo ora ad analizzare un altro metodo di indagine, che consiste nell'effettuare l'analisi a livello di documento (*document level sentiment classification*). Al contrario di quanto avviene nel primo caso, questa modalità di indagine prevede che l'intero documento diventi l'informazione base su cui lo studio si sviluppa e venga quindi classificato, ottenendo l'assegnazione di una polarità che può essere positiva o negativa.

La maggior parte delle ricerche che si basano su questo tipo di approccio utilizza come dati le recensioni di prodotti: si ritiene importante, quindi, fornire una definizione riguardo al contesto delle recensioni, tenendo tuttavia presente che il medesimo concetto può essere applicato anche ad altri ambiti.

Definizione 1.2.1. *Document Classification*: dato un documento d che esprime un'opinione riguardo un'entità, deterimarne il sentimento s dell'*opinion holder* riguardo l'entità. Ovvero, in maniera più formale, determinare un'opinione s espressa nell'*aspect* GENERAL nella quintupla

$$(\_, GENERAL, s, \_, \_), \tag{1.1}$$

dove l'entita e, l'*opinion holder* h e l'istante di tempo dell'opinione t si assumono essere irrilevanti.

Si possono formulare, inoltre, due diverse definizioni a seconda di come viene espressa l'opinione s. Se s assume valori binari, come positivo o negativo, allora il problema si definisce di classificazione. Se s, invece, assume valori

numerici o valori ordinali entro un deteriminato *range*, come, per esempio, valori da 1 a 5, il problema si definisce di *regression*.

Per rendere questa definizione applicabile a problemi reali, le ricerche esistenti assumono la seguente ipotesi come vera (Ding and Liu, 2010)

Definizione 1.2.2. La *sentiment classification o regression* assume che l'opinione del documento d esprime opinioni riguardo una singola entita e e contiene opinioni da un singolo *opinion holder* h.

Nella pratica, però, se il documento valuta più di una entità, allora le opinioni espresse su queste differenti entità possono essere diverse, eterminando opinioni positive per alcune entità e negative per altre: per tale motivo questo approccio non sempre è utilizzabile nella realtà. Nel caso delle recensioni, tuttavia, questo metodo è stato largamente usato, in quanto si tratta di valutazioni che, nella maggior parte dei casi, si incentrano su un singolo prodotto o su un servizio specifico e sono scritte da un unico autore; se, al contrario, prendiamo in considerazione un blog o i messaggi scritti su un forum possiamo notare che le opinioni espresse dagli autori riguardano, generalmente, più entità e che le frasi utilizzate mettono queste entità a confronto.

1.2.1 Ricerca delle Feature

Come vedremo nel paragrafo 1.4, prima di procedere alla classificazione vera e propria mediante l'applicazione di varie metodologie, è necessario mettere in atto dei sistemi per l'estrazione delle feature, quelle parti di testo, cioè che verranno utilizzate per stabilire la polarità del testo stesso. Alcuni esempi di *feature* utilizzate sono i seguenti:

- *Termini e la loro frequenza:* le parole da utilizzare come *feature* possono essere prese da un dizionario di *opinion words* come vedremo in 1.4 . Un altro possibile sistema può essere quello di identificare i termini con le parole dell'insieme dei documenti presi a riferimento, esaminando, ad esempio, tutte le parole che compaiono almeno un certo numero di

volte nei documenti. Scelte più raffinate, infine, prevedono di associare dei termini a gruppi di parole o n-grammi.

- *Part of speech (POS)*: parole appartenenti a differenti parti del discorso possono essere trattate in maniera differente. È stato dimostrato, ad esempio, che gli aggettivi sono importanti indicatori di opinioni, per cui vengono utilizzati come *feature* in svariate ricerche. Anche i modi di dire (*opinion phrase*) di una particolare lingua sono evidentemente utili per la rilevazione del *sentiment*.

- *Sentiment words:* si tratta di parole che vengono utilizzate per esprimere un sentmiento positivo o negativo. Le parole "good", "wonderful", "amazing", ad esempio, sono *sentiment word*, positive mentre parole come "bad, poor e terriblè' sono *sentiment word* negative.

- *Sentiment shifter*: sono espressioni usate per cambiare l'orientamento dei sentimenti. I principali *sentiment shifter* sono costituiti da parole di negazione, come possiamo vedere, per esempio, nella seguente frase "I don't like this camera ".

1.3 Sentence Sentiment Classification

Come già evidenziato nel paragrafo precedente, effetuare *sentiment classification* a livello di documenti potrebbe costituire un approccio troppo grossolano per alcune applicazioni. Per questo motivo si ritiene utile descrivere, di seguito, una diversa modalità di analisi, a livello di frase: si tratta, cioè, di effettuare una classificazione di sentimento per ogni singola frase. A livello concettuale, non ci sono differenze tra le due metodologie, in quanto l'analisi dei sentimenti a livello di frase può essere eseguita come se le frasi stesse fossero interi documenti, applicando quindi le stesse tecniche presentate per la *document classification*: anche questo tipo di approccio, infatti, parte dal presupposto che ad ogni frase sia associata una sola, singola opinione. Anche in questo caso, non sempre nella realtà è possibile rispettare tale assunto, ma

si può facilmente intuire che, essendo applicato a livello di frase, è molto più probabile che sia vero rispetto a quello applicato ad un documento.

Definizione 1.3.1. Problema di *Sentence classification*: data una frase x, determinare se x esprime un'opinione positiva, negativa o neutra (cioè nessuna opinione).

In questo contesto, la definizione basata sulla quintupla (e, a, s, h, t) non viene usata, perché la classificazione a livello di frase impiegata sempre come passo intermedio: quando si applicano le tecniche di *sentence sentiment classification*, infatti, bisogna già conoscere su quali *target* viene espressa l'opinione.

Il problema di *sentiment classification* a livello di frase può essere affrontato in due modi differenti: come problema di classificazione multiclasse oppure come se si trattasse di due problemi di classificazione binaria separati. Nel secondo caso, il primo problema di classificazione (anche chiamato primo *step*) è quello di determinare se la frase esprime un'opinione, mentre il secondo problema (secondo *step*) consiste nel classificare se, nelle frasi che esprimono opinione, questa sia positiva o negativa. Il primo problema presentato viene generalmente chiamato *subjectivity classification*: si tratta, cioè, di determinare se una frase esprime un'informazione soggettiva o oggettiva (Yu and Hatzivassiloglou, 2003; Hatzivassiloglou and Wiebe, 2000; Riloff and Wiebe., 2006) Alle frasi oggettive, in genere, non viene attribuito nessun tipo di sentimento o opinione. In certi casi, tuttavia, anche questo approccio può risultare problematico perché a volte le frasi oggettive esprimono opinioni. Tale concetto può essere esemplificato da una recensione che contenga la frase "Ha smesso di funzionare ieri ": frase chiaramente oggettiva, ma esprime, tuttavia, un'opinione negativa, perché, se applicata ad un prodotto, esplicita una situazione non desiderata.

1.3.1 Periodi ipotetici

La maggior parte delle ricerche sulla *sentence sentiment classification* sono incentrate sulla risoluzione a livello generale di questo problema senza approfondire né tenere conto dei diversi tipi di frasi e il fatto che alcuni tipi di frase possono richiedere diversi tipi di approcci. Come emerge dalle ricerche di Ramanathan et al. (2009), difficilmente si può trovare un approccio che funzioni correttamente e con buoni risultati per ogni tipo di frase, ma è necessario focalizzarsi su ciascuna di esse e scegliere per ognuna le tecniche più adatte. Una tipologia di enunciato molto studiata riguarda le *conditional sentences* (periodo ipotetico), che esprimono un parere riguardo situazioni ipotetiche e le relative conseguenze. In questo caso ci troviamo di fronte a espressioni composte da due parti dipendenti fra di loro: la *condition clause* e la *consequent clause*. Si citano, di seguito, due esempi di periodi ipotetici sono i seguenti "If someone make a reliable car, I will buy it" o anche "If your Nokia phone is not good, buy this Samsung phone.". Analizzandoli, si può notare come un approccio normale potrebbe portare a risultati fuorvianti. Nel primo esempio non viene espresso nessun tipo di opinione riguardo alle macchine, anche se il termine "reliablè' può essere annoverato fra le parole che solitamente esprimono un'opinione positiva; nella seconda frase, invece, potremmo riconoscere un'opinione positiva riguardo il telefono Samsung ma nessuna opinione riguardo al Nokia.

1.3.2 Frasi sarcastiche

Un'altra problematica che può essere molto difficile da affrontare riguarda le frasi sarcastiche, cioè quelle in cui l'autore esprime l'opposto di ciò che in realtà vuole dire. Il sarcasmo è stato studiato in liguistica, in psicologia e in scienze cognitive da (Gibbs and Raymond, 2015; Gibbs et al., 2007; Kreuz et al., 2007, 1989; Akira, 2000), cosi come nel contesto della *sentiment analysis* (González-Ibáñez et al., 2011): si tratta, infatti, di una tematica particolarmente complessa, data la grande difficoltà nel trovare metodologie

e sistemi in grado di distinguere frasi sarcastiche da frasi che esprimono opinioni sincere.

1.4 Tecniche di Sentiment Classification

Gli approcci utilizzati per effettuare *Sentiment Classification* si possono dividere in base a tre diverse metodologie: l'approccio tramite machine learning (ML), quello basato sul Lexicon e infine un approccio ibrido che utilizza ambedue le tecniche.

Il metodo basato sul machine learning si focalizza nell'utilizzo di algoritmi già impiegati nel settore della *text classification*, affinandoli, però, grazie all'uso di informazioni estrapolate dalle features linguistiche. L'approccio basato sul *lexicon* ricorre ad un *sentiment lexicon*, cioè su un insieme di termini che esprimono informazioni riguardanti un particolare *sentiment*, già conosciuti ed elaborati prima di iniziare l'analisi. Questo tipo di metodologia può ricorrere a due diverse tecniche, e cioè di un approccio *dictionary based* e in uno *corpus based*: quest'ultimo, in particolare, prevede la valutazione della polarità tramite l'utilizzo di metodologie statistiche o semantiche. L'approccio ibrido, infine, molto diffuso, combina entrambe le metodologie precedentemente citate, utilizzando spesso il *sentiment lexicon* come chiave principale per l'analisi.

I sistemi di classificazione che si servono dell'approccio ML possono essere divisi in due ulteriori categorie, a seconda che usino metodologie supervisionate o che si basino su una metodologie non supervisionate. Generalmente, una metodologia supervisionata ha risultati migliori rispetto alla seconda che, invece, viene spesso usata quando si necessita di una grande mole di dati o quando risulta impossibile accedere a dataset etichettati. I sistemi basati sul *lexicon*, invece, sono incentrati sulla ricerca di una *opinion lexicon* che serve per analizzare il testo. I metodi a cui si fa ricorso sono generalmente due: il primo è basato sul dizionario e, utilizzando parole chiave (*seed words*) di polarità nota, effettua una ricerca dei sinonimi e degli antonimi delle suddette

parole tramite il dizionario stesso. Il secondo metodo, invece, è quello definito *corpus based* e, partendo sempre dall'utilizzo di *seed words*, utilizzando *pattern sintattici* per ricercare in un corpus altre *opinion words*.

1.4.1 Approccio basato su Machine Learning

L'approccio di Machine learning si serve di algoritmi già utilizzati nel ML per risolvere il problema di sentiment analysis, rapportandolo ad un problema classico di text classification in cui vengono impiegate *features* sintattiche e/o linguistiche. Come spiegato nel lavoro di Melloncelli (2012) possiamo generalizzare l'approccio ML con la seguente definizione: il testo da classificare può essere definito da un insieme di m valori reali (anche detti *features*); ad ogni istanza del problema viene associata un'etichetta scelta in un insieme di possibili etichette E; l'algoritmo richiede in input il *training set* $S = \langle x_i, y_i \rangle | i \in (0 \cdots n)$ dove $x_i \in R^m$ (esempio di oggetto da classificare) e $y_i \in E$ (etichetta ad esso associata); alla fine viene prodotta una funzione di modello $\hat{y}_i = f(x_i)$ che deve essere in grado di massimizzare il numero di oggetti classificati correttamente, ovvero per cui $y_i = \hat{y}_i$.

Supervised learning

L'apprendimento basato sul metodo supervisionato ricorre all'uso di documenti etichettati, cioè documenti sui quali è stata effettuata una annotazione manuale: questi testi sono usati per svolgere il training del sistema. L'approccio supervisionato prevede l'utilizzo di numerosi tipi di classificatori, i più diffusi dei quali vengono di seguito presentati brevemente.

Naive Bayes Classifier (NB): il classificatore NB è il classificatore più semplice e per questo è anche quello che viene utilizzato più comunemente: questo tipo di classificatore calcola la probabilità a posteriori di una determinata classe, basandosi sulla distribuzione di parole nel documento. L'estrazione delle *feature* dal documento viene fatta generalmente tramite *BOW* un modello che rappresenta il testo trattato come insieme (*bag*) di

parole. Utilizzando un esempio possiamo meglio spiegare il funzionamento di un modello BOW: consideriamo le seguenti frasi:

```
(1) John likes to wtach movies. Mary likes movies too.
(2) Jhon also likes to watch football games.
```

Successivamente viene costruita una lista composta dalle parole utilizzate nelle due frasi considerate:

```
[
    "John",
    "likes",
    "to",
    "watch",
    "movies",
    "also",
    "football",
    "games",
    "Mary",
    "too"
]
```

Utilizzando gli indici di questa lista è possibile comporre due vettori:

```
(1) [1, 2, 1, 1, 2, 0, 0, 0, 1, 1]
(2) [1, 1, 1, 1, 0, 1, 1, 1, 0, 0]
```

Infine queste due liste rappresentano le frasi analizzate, in particolare vediamo coma la lista (1) abbia come primi due elementi i valori 1 e 2. Il valore 1 specifica che la parola "jhon" è presente nel documento (1) una sola volta mentre il 2 rappresenta il fatto che la parola "likes" invece appare due volte. È facile capire che questo sistema non consideri la posizione delle parole nella sua analisi. Per quanto riguarda l'elemento discriminante l'NB utilizza il teorema di Bayes per calcolare la probabilità che una determinata *feature* appartenga ad una etichetta.

$$P(label|feature) = \frac{P(label) \cdot P(features|label)}{P(features)} \tag{1.2}$$

Dove $P(label)$ sia la probabilità a priori, $P(features|label)$ è la probabilità a priori che un una determinata *feature* sia classificata con una specifica etichetta, infine $P(feature)$ è la probabilità a priori che ci sia una occorrenza di una data *feature*. Utilizzando l'ipotesi Naive possiamo dire che tutte le *features* sono indipendenti e l'equazione può essere riscritta come segue:

$$P(feature|label) = \prod_{i=1}^{n} P(feature_i|label) \tag{1.3}$$

Nelle ricerche di Hanhoon et al. (2012) possiamo vedere come sia stato implementato un classificatore NB migliorato per risolvere la tendenza che l'accuracy della classificazione positiva risulti il 10% più elevata rispetto alla all'accuracy di quella negativa.

Bayesian Network (BN): il presupposto da cui parte il sistema di classificazione NB riguarda l'indipendenza delle *feature* insieme all'assunto in base al quale tutte le *feature* sono completamente indipendenti. Il modello tramite *Bayesian Network*, invece, utilizza un grafo diretto aciclico, dove i nodi rappresentano le variabili e i vertici le dipendenze condizionali: questo comporta che un BN rappresenti in modo completo le variabili e le loro relazioni. Come possiamo leggere da Loeu et al. (2013) la complessità computazionale di un modello BN è estremamente elevata e questo limita fortemente le applicazioni di questo tipo di classificatore. Nonostante ciò, Jonathan et al. (2012) hanno dimostrato come un metodo semi-supervisionato che utilizza un classificatore BN possa essere usato con risultati notevoli su un compito di classificazione multi-dimensionale.

Maximum Entroy (ME): un classificatore ME (conosciuto anche come un *conditional exponential classifier*) converte gli insiemi di *feature* in vettori. Questi vettori vengono utilizzati per calcolare i pesi di ogni *feature*, i quali, infine, vengono combinati per stabilire come etichettare un determinato set di *feature*. Questo classificatore viene parametrizzato con un insieme $X(weights)$ utilizzato per calcolare le *feature* generate dalle stesse attraverso

un $X(encoding)$. In particolare l'enconding effettua un *mapping* tra le coppie $C(featureset, label)$ con un vettore. La probabilità di ogni etichetta viene quindi calcolata nel seguente modo:

$$P(fs|label) = \frac{weights \cdot encode(fs, label)}{\sum_{l \in labels}(weights \cdot encode(fs, l)} \tag{1.4}$$

Kaufmann (2012) dimostra come utilizzare un classificatore ME per effettuare un parallelismo tra frasi di due lingue, servendosi di un insieme di dati di *training* molto piccolo.

Support Vector Machines (SVM): dato $X = (x_1, \ldots, X_n)$ come la frequenza normalizzata delle parole in un documento, il vettore $A = (a_1, \ldots, a_n)$ rappresenta un insieme di coefficienti lineari con la stessa dimensione delle *feature*; con il termine b si intende uno scalare, mentre l'output di un classificatore lineare è definito come $p = A.X + b$ e utilizza l'iperpiano come separazione tra differenti classi. Nel modello SVM si ricercano questi separatori lineari in modo che vengano separate le diverse classi che si vogliono suddividere con la maggior precisione possibile. I testi, in genere, si adattano molto bene per una classificazione SVM, grazie alla natura sparsa del testo, dove alcune *feature* sono irrilevanti ma tendono ad essere correlate fra loro e generalmente organizzate in categorie ben separabili. Gli utilizzi di questo tipo di classificatore sono numerosi, Chien and You-De (2011) dimostrano come servirsi di classificatori SVM per valutare la polarità di recensioni di prodotti, Hu and Wenjie (2011) dimostrano, invece, come non solo sia possibile effettuare una classificazione binaria sulla polarità espressa dai messaggi di Twitter, ma anche come si possa effettuare un'analisi più approfondita , assegnando un punteggio su quanto sia positivo o negativo un particolare messaggio.

Neural Networks(NN): si tratta di una rete neurale formata da numerosi neuroni, dove con neurone si intende l'unità base della rete. L'input dei neuroni è denotato con un vettore $\vec{X_i}$ che rappresenta la frequenza delle parole in un particolare documento. Ad ogni neurone è associato un peso A usato nella funzione $f()$ che elabora gli input, e la funzione lineare della

rete neurale è $P_i = A \cdot \vec{X}_i$ in un problema di classificazione binaria il segno della funzione p_i determina a quale classe appartenga l'input. Alcune ricerche pubblicate da Rodrigo et al. (2013) dimostrano come una NN abbia risultati migliori di SVM su numerosi problemi di classificazione: in particolare sono stati effettuati test sulla classificazione di commenti su film, recensioni di prodotti elettronici e di libri.

Rule-Based classifier: nei classificatori *rule based* i dati vengono modellizzati come un insieme di regole. Il lato sinistro della rapresentazione esprime una condizione sul *feature set* espressa in forma normale disgiuntiva, mentre il lato destro assegna un'etichetta. Queste regole vengono generate durante la fase di test tramite determinati criteri; i più comuni sono definiti *support* e *confidence*. Il primo definisce il numero assoluto di istanze nel testo utilizzato come *training* rilevanti per una determinata regola; il secondo, invece, definisce la probabilità condizionata che il lato destro della regola sia soddisfatto quando anche il lato sinistro lo sia.

Weakly, semi e unsupervised learning

L'obiettivo principale della classificazione di un testo è quello di suddividere un documento in categorie predefinite. Come visto finora, per svolgere questo compito si utilizza un numero elevato di documenti di *training*, ma spesso la creazione e la raccolta di questi dati etichettati può essere particolarmente difficile, mentre la ricerca di documenti non etichettati in genere risulta essere estremamente più semplice. In numerose ricerche, come quelle presentate da Youngjoong and Jungyun (2000), per aggirare queste difficoltà sono stati proposti metodi in cui un testo viene diviso in frasi che vengono categorizzate utilizzando liste di *keyword* per ogni categoria ricorrendo alla *similarity* tra le varie frasi. Yulan and Deyu (2011), al contrario, hanno proposto una strategia di *weak supervision* a livello di *feature*, utilizzando un classificatore iniziale contenente informazioni estratte da un *sentiment lexicon* esistente in un modello di classificazione. Queste *feature* già etichettate vengono utilizzate come vincolo per effettuare una previsione sui dati non

etichettati impiegando un *generalized expectation criteria*. Nel loro lavoro vengono individuate parole che esprimono polarità in un dominio specifico e viene dimostrato come queste parole possano esprimere una diversa polarità in altri domini.

1.4.2 Approccio basato sul Lexicon

Le *opinion word* sono utilizzate in molti *task* di classificazione di sentimenti: queste parole, insieme a frasi che esprimono opinioni e idiomi, sono definite *opinion lexicon*. Per la raccolta e la categorizzaione di queste *opinion word* si fa ricorso a tre approcci principali: il primo, quello manuale, non viene mai usato da solo poiché è molto lento, ma viene impiegato insieme agli altri due approcci automatici, il *dictionary-based* e *corpus-based*.

Dictionary-based Approach

Come possiamo leggere da Kim and Hovy (2004) la strategia impiegata in un approccio *dictionary based* è quella di ottenere e categorizzare manualmente un piccolo insieme di *opinion word*; questo insieme viene successivamente ampliato, ricercando sinonimi e antonimi di queste parole, ricorrendo a corpora quali *WordNet* e *Thesaurus*. Le nuove parole ottenute in tal modo vengono aggiunte all'insieme di parole iniziali; riparte quindi una nuova iterazione, finchè non vengono trovate altre parole da aggiungere all'insieme. Il principale svantaggio di questo tipo di approccio è quello di non riuscire ad ottenere *opinion words* riguardanti specifici domini o contesti. La ricerca di **?** dimostra come questo tipo di metodo sia particolarmente efficace per la ricerca e l'estrazione di parole chiave relative alle pubblicità online e nel generare una strategia pubblicitaria più rilevante per il consumatore.

Corpus-based Approach

L'approccio *corpus-based* viene utilizzato per cercare di ottenere *opinion words* relative ad un particolare contesto. Vengono usati *pattern* sintattici

insieme a delle *seed words* per ricercare in un corpus altre *opinion words*. Uno di questi metodi è presentato nelle ricerche di Hatzivassiloglou and McKeown (1997) dove, partendo da una lista di *seed words*, in particolare di aggettivi che esprimono una opinione, è stato utilizzati un insieme di vincoli sintattici, nella forma di connettivi del tipo AND, OR, BUT, etc. Per esempio: la congiunzione AND esprime il fatto che gli aggettivi congiunti hanno la stessa polarità. Questa idea è definita *sentiment consistency*, ma non sempre può essere utilizzata in pratica. Allo stesso modo le espressione avversative come BUT o HOWEVER definiscono un cambiamento di opinione. Un altro approccio, basato su *Conditional Rando Fields (CRF)*, è quello utilizzato dalla ricerca di Jian and Yanquan (2011) tale metodo si basa su un algoritmo di *multi-string matching* utilizzato per discriminare la polarità di una frase ed è stato usato con successo per categorizzare varie tipi di recensioni (su auto, hotel e computer). Cheng-Yu et al. (2010) forniscono un ulteriore approccio che rientra nella classificazione *corpus based*: si tratta del metodo tassonomico, in cui le *feature* vengono mappate in una *feature taxonomy* che raffigura una rappresentazione semantica delle parti e degli attributi contenenti opinioni di un oggetto. Il sistema proposto dai due studiosi è *domain-oriented*, ed in esso si trovano definite le risorse specifiche che vengono utilizzate sul dominio per catturare informazioni sul modo in cui le persone esprimono opinioni al suo interno (in questo caso i domini trattati sono quelli delle recensioni di cuffie musicali, hotel e automobili). Uno dei principali risultati di questo studio è stato quello di dimostrare come, in certi ambiti, un modello specializzato su particolari domini fornisca risultati migliori di altri modelli *domain indipendent*.

Approcci statistici: si possono trovare pattern di co-occorrenza o *seed words* utilizzando tecniche che si servono di funzionalità statiche. Come proposto da Fahrni and Klenner (2008) è possibile utilizzare un insieme di corpus annotati raccolti nel web per la costruzione del dizionario: in questo modo si risolve il problema dovuto al fatto che la creazione di un dizionario da un solo corpus relativamente alle parole che non vengono utilizzate all'interno

dello stesso può essere difficoltosa (sopratutto se questo corpus è di dimensioni ridotte). La polarità di queste parole può essere ottenuta studiando la frequenza di occorrenza in un corpus annotato di dimensioni adeguate. Se una parola è presente più volte in testi aventi una annotazione positiva le verrà attribuito un valore positivo, se invece la frequenza è negativa il valore attribuito sarà a sua volta negativo. Un'altra modalità utilizzata per determinare la polarità di una parola è confrontare la co-occorrenza rispetto a parole di polarità nota: se in un contesto una determinata parola si presenta con una notevole frequenza è allora altamente probabile che la polarità di entrambe coincida. Questi metodi sono utilizzati nelle ricerche di Nan et al. (2012) che, analizzando lo stile di scrittura di recensioni ritenuto dagli autori casuale visti e considerati i vari background dei consumatori, attraverso un'analisi statistica sono giunti alla conclusione che il 10% delle recensioni su Amazon sono soggette a manipolazioni.

Approccio semantico: l'approccio semantico fornisce un valore di polarità basandosi sui principi di *similarity* tra parole, stabilendo una polarità simile a parole semanticamente simili. Se condiseriamo *WordNet*, il più grande database semantico-lessicale per la lingua inglese, vediamo come l'organizzazione dei termini per significato affine, chiamati *synset*, e dai collegamenti dei loro significati attraverso diversi tipi di relazione, può essere utilizzata per ottenere una lista di *sentiment words* effettuando una iterazione in un set di parole iniziale ed espandendo questo insieme: stabilendo la polarità di parole nuove basandosi sul numero di sinonimi e antonimi, una parola avente molti sinonimi con polarità positiva viene percepita anch'essa come positiva. Come presentato da Isa and Piek (2012) questo metodo viene utilizzato in numerose applicazioni per costruire un *lexicon* che, partendo da verbi, nomi e aggettivi, viene poi usato in *task* di *sentiment analysis*. Un'applicazione pratica la vediamo nel lavoro pubblicato da **?** che abbina un approccio semantico ad uno statistico per ricercare le debolezze di particolari prodotti partendo dalle recensioni online.

Lexicon-based e tecniche NLP

Tecniche di NLP possono essere utilizzate, in genere insieme ad un approccio basato sul *lexicon*, per trovare la struttura sintattica del documento analizzato e per poter individuare le relazioni semantiche. L'approccio proposto da Francis (2012) utilizza un *dependency parsing* come passo di preprocessing per un algoritmo definito di *sentiment propagation*. I due ricercatori assumono che ogni elemento linguistico - come i nomi, i verbi etc. - abbia un valore intrinseco di sentimento, che viene propagato attraverso la struttura sintattica della frase esaminata. Il lavoro di Hye-Jin and Jong (2012), invece, utilizza l'analisi NLP da una differente prospettiva, usando tecniche per identificare le coniugazioni verbali e le espressioni di tempo: in questo modo, grazie anche ad un algortimo di *ranking*, vengono categorizzate recensioni di prodotti in differenti periodi di tempo.

Un'analisi effettuata da Asher et al. (2008) dimostra come le informazioni del discorso possano essere utilizzate per effettuare *sentiment annotation*. In particolare sono state individuate cinque tipi di relazioni retoriche, *Contrast, Correction, Support, Result e Continuatum*, a cui varie informazioni di *sentiment* sono state assegnate per poi essere utilizzate per l'annotazione.

Attraverso la *Rhetorical Structure Tehory (RST)*, che descrive come suddividere il testo in diverse parti, ciascuna delle quali rappresenta una parte sensata del testo, Heerschop et al. (2011) hanno proposto un framework che utilizza questa metodologia a livello di frase per effettuare una *sentiment analysis* basata a livello di struttura. In questo modo è stato categorizzato il testo in due classi distinte, la prima contenente informazioni importati, che verranno utilizzate per esprimere una polarità nella frase, e l'altra meno importante, contenente una parte di testo che verrà invece viene ignorato. Questa tecnica viene usata con successo anche da Chenlo et al. (2013) dove RST viene applicata su post di blog e se ne servono per estrarre frasi relative all'argomento del post. Queste frasi vengono poi analizzate in modo da ottenere la polarità per l'intero testo.

1.4.3 Altre tecniche

Esistono altre tecniche che sono tuttavia difficilmente riconducibili ai due grandi insiemi presentati finora (ML e Lexicon-based): una di queste, proposta da Wille (1982) viene definita *Formal Concept Analysis (FCA)*. e consiste in un un approccio matematico basato sulle connessioni di Galois, usato per definire la struttura, analizzare e visualizzare i dati. Questi dati sono composti da una lista di entità con le relative *feature* e sono strutturati in astrazioni formali chiamate *formal concepts*. Tali strutture formano un reticolo completo parzialmente ordinato, definito come *concept lattice*. Questi reticoli vengono costruiti individuando gli oggetti e i rispettivi attributi relativi ad uno specifico dominio, attraverso un sistema di analisi che tiene conto anche delle eventuali informazioni non certe o non chiare. Il sistema fornisce un elenco di relazioni tra i vari oggetti. Nell'ambito della *sentiment analysis* la tecnica FCA viene applicata in diversi ambiti, come mostrato da Li and Tsai (2011) che hanno effettuato una classificazione usando *concept* anziché documenti: questa tecnica riduce le incertezze e aumenta la precisione dei risultati quando vengono analizzati termini ambigui. Dai risultati si evince anche come questa tecnica riduca la sensibilità alle interferenze presenti nei testi e aumenti inoltre l'adattabilità nell'utilizzo di questi sistemi in caso di applicazioni di *cross domain*.

È interessante notare come la tecnica FCA è stata utilizzata con successo anche per costruire un modello di dominio ontologico, come dimostrato da uno studio di Efstratios et al. (2013) che presenta tecniche basate su ontologie per fornire una analisi più accurata relativa ai sentimenti espressi su Twitter; per raggiungere tale scopo lo studioso ricorre alla suddivisone dei Tweet in un inseme di *feature* che risultano rilevanti rispetto all'argomento cercato.

1.5 Valutazione del risultato dei classificatori

La valutazione prestazionale dei classificatori di testi è effettuata sperimentalmente piuttosto che analiticamente, in quanto una valutazione di tipo

analitico non può essere formalizzata (a causa della natura soggettiva del problema della classificazione). La valutazione sperimentale di un classificatore solitamente misura la sua efficacia, ovvero l'abilità di prendere la giusta decisione durante il processo di classificazione.

In genere, per valutare il risultato di un classificatore viene utilizzata una *confusion matrix*, cioè una particolare matrice che permetta la visualizzazione del risultato di un classificatore. Ogni colonna della matrice rappresenta le istanze della classe assegnata dal classificatore, mentre le righe rappresentano la classe reale (o viceversa).

		Risultato Previsto		
		p	**n**	**totale**
Valore reale	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	totale	P	N	

Tabella 1.1 Esempio di *confusion matrix*, si può vedere come vengono rappresentati i vari valori rispetto al valore reale e a quello previsto dal categorizzatore

I valori utilizzati nella *confusion matrix* sono i seguenti:

- True Positive (TP): classificato correttamente
- True Negative (TN): correttamente rifiutato
- False Positive (FP): falso allarme, errore di tipo I
- False Negative (FN): mancato riconoscimento, errore di tipo II

I valori utilizzati per la valutazione del risultato sono l'accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.5}$$

La precision:

$$Precision = \frac{TP}{TP + FP} \tag{1.6}$$

Il recall:

$$Recall = \frac{TP}{TP + FN} \tag{1.7}$$

e l'F-Score:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{1.8}$$

Questi valori aggiuntivi sono necessari perché l'accuracy da sola non fornisce un buon metro di paragone per la valutazione delle performance. Il valore riportato dall'accuracy, infatti, può essere ingannevole nel caso in cui l'insieme di dati sia sbilanciato (qualora cioè, il numero di campioni tra le classi analizzate sia molto differente): per esempio nel caso voglia classificare un inseme composto da 95 entità di tipo A e 5 di tipo B il classificatore potrebbe funzionare in maniera imparziale e classificare tutte le entità assegnandogli il tipo A. In questo caso l'accuracy risulterebbe del 95% mentre il vero valore di riconoscimento sarebbe il 100% per la classe A e 0% per la classe B.

In un processo di classificazione statistica, la *precision* per una classe è il numero di veri positivi (il numero di oggetti etichettati correttamente come appartenenti alla classe) diviso il numero totale di elementi etichettati come appartenenti alla classe (ottenuto dalla somma di veri positivi e falsi positivi, questi ultimi consistenti in oggetti etichettati erroneamente come appartenenti alla classe). In questo contesto, il *recall* è definito come il numero di veri positivi diviso il numero totale di elementi che attualmente appartengono alla classe (per esempio la somma di veri positivi e falsi negativi, ovvero oggetti che non sono stati etichettati come appartenenti alla classe ma dovrebbero esserlo).

Infine con il termine *F-score* si intende la misura della precisione del test. Nella sua definizione viene considerata sia la *precision* che il *recall* e il

risultato ottenuto rappresenta la media armonica di questi due valori. Con il valore 1 viene rappresentato il valore migliore mentre con 0 quello peggiore.

1.6 Modelli utilizzati

Come già esposto nel Capitolo e come si può desumere dalle ricerche effettuate da Bo and Lee (2008) la *sentiment classification* è uno degli aspetti più studiati della NLP. L'obiettivo che questo tipo di analisi si prefigge è quello di operare una classificazione su determinati testi, suddividendoli in base ai contenuti, positivi oppure negativi, che gli stessi esprimono.

Una ricerca condotta da Zhai et al. (2010) ha portato a teorizzare due diverse modalità per affrontare questo problema: la prima, denominata *classification* categorizza il testo in base a due valori differenti (es. positivo o negativo); la seconda, definita *regression*, invece assegna un punteggio ai testi analizzati (es. un valore da 1 a 5) in modo tale che la loro differenziazione si basi su un maggior numero di elementi discriminanti.

Entrando nello specifico, possiamo constatare come il lavoro di Liu et al. (2007) proponga una *sentiment analysis* per ipotizzare le performance di vendita di una particolare azienda. È interessante, inoltre, soffermarsi sulle numerose ricerche svolte nell'ambito dei social network, settore che oltretutto è più vicino all'argomento trattato in questa tesi: ricordiamo, in primo luogo, lo studio Tae and Smith (2010) che presenta un metodo per prevedere l'esito di film, analizzando i dati provenienti dai forum e dal sito web *Internet Movie Database*; è indicativo, inoltre, il lavoro di Georg and Hauffa (2011) che analizza l'evolversi delle relazioni sociali tra utenti di Twitter.

Un ulteriore e interessante contributo a questo argomento viene fornito dalla pubblicazione di Mesnil et al. (2015), dove vengono esaminati vari approcci di *machine learning* al problema di *sentiment analysis*, in particolare viene presentato l'approccio tradizionale, che utilizza tecniche di *bag-of-words* o di *bag-of-ngram* (Christopher et al., 2008; Bo and Lee, 2008) e, in dettaglio, un approccio più complesso che impiega un *language model* generativo

abbinandolo ad un sistema discriminativo: l'utilizzo di quest'ultimo metodo, che usa in maniera simultanea due tipi di modelli complementari aumenta la precisione del risultato. La logica sottesa al modello generativo consiste nel fatto che, dopo aver effettuato il *training* di due modelli su dati che esprimono pareri positivi e pareri negativi, si utilizza la percentuale di somiglianza di questi due modelli rispetto ai dati del file di test. In questo modo possiamo dunque ipotizzare che sia maggiore la possibilità che uno scritto connotato da parere positivo tra quelli utilizzati come test, venga generato da un modello proveniente da un set di dati che esprimono pareri positivi rispetto a quelli che esprimono pareri negativi.

In questa tesi è stato utilizzato proprio questo approccio, e cioè l'impiego di un sistema generativo in concomitanza con un sistema discriminativo: in particolare, come modello generativo è stata usata una *recurrent neural networks* e nello specifico RNNLM e, come sistema discriminativo *word2vec*.

Alcune ricerche svolte da Mikolov (2012) sull'argomento ci confermano, per quanto riguarda il *word embedding*, che gli *statistical language models* sono ampiamente utilizzati in diverse applicazioni, come per esempio la *speech recognition* o *machine translator*. Tradizionalmente le tecniche per la costruzione di questi modelli si basano sull'utilizzo di N-grammi che, pur avendo evidenziato alcune debolezze e nonostante l'enorme sforzo della comunità di ricerca in numerosi campi (*speech recognition*, neuroscienza, intelligenza artificiale) per trovare un modello alternativo sono rimasti per lungo tempo come lo stato dell'arte. Recentemente, tuttavia, è stato possibile costruire *Language Models*(LM) tramite *Recurrent Neural Network*(RNN), come mostrato da Mikolov et al. (2010, 2011a), è stato altresì dimostrato come queste RNN LM abbiano una qualità superiore rispetto alle tecniche che utilzzano *ngrams* anche se ciò comporta una più alta complessità computazionale.

Proprio analizzando questo modello basato su RNN, per la precisione quello proposto nel lavoro pubblicato da Mikolov et al. (2011b), appare evidente come sia stato possibile ridurre la complessità di calcolo e realizzare RNN LM con velocità superiori (anche fino a quindici volte) rispetto a RNN

base attraverso l'utilizzo di un algoritmo di *backpropagation throught time.*

Per quanto riguarda il modello discriminativo, in letteratura possiamo vedere come il lavoro di Baroni and Lenci (2010) dimostri in che modo una rappresentazione di parole distribuita, basata sulla *Harris distributional hypotesis*, abbia giocato un ruolo centrale nel campo dell'analisi del linguaggio naturale (NLP) più recentemente in vari studi proposti da Collobert et al. (2011); Mikolov et al. (2013a,b) sono stati utilizzati vettori densi derivati da RNN. Questi vettori vengono più specificatamente definiti *word embedding* e sono stati adottati in numerosi tasks NLP.

Il lavoro presentato è basato sulle ricerche effettuate da Mikolov et al. (2013b) che presentano due differenti modelli di *word embedding*, CBOW e skip-gram trattati più approfonditamente nel paragrafo 1.8: entrambi i modelli, attraverso l'uso della regressione lineare, prevedono la parola target. In particolare, come presentato nella ricerca di Le and Mikolov (2014) è stato impiegato un modello che utilizza i *paraghrap vectors*(PV) trattato nel paragrafo 1.8.4.

1.7 Recurrent Neural Networks

Gli esseri umani non iniziano a pensare da zero: esiste infatti per la mente umana una capacità di associare tra loro informazioni passate e recenti che è stata definita *persistenza della memoria.* Il lettore di questa tesi, ad esempio, assocerà il significato di ogni parola basandosi su quelle lette in passato. Le reti neurali tradizionali, al contrario, ignorano gli eventi passati. Volendo prendere ad esempio la classificazione della successione delle scene di un film, non è possibile che una rete neurale possa utilizzare gli eventi passati per classificare quelli correnti.

Per cercare di risolvere questo problema sono state create le *recurrent neural networks (RNN)*: a differenza delle reti neurali tradizionali le *RNN* hanno dei cicli interni che permettono all'informazione di persistere.

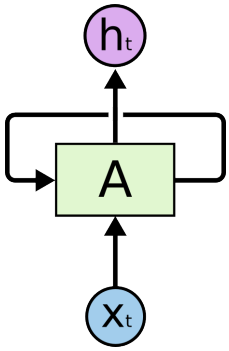

Figura 1.1 Ciclo interno di una RNN

Nella figura 1.1, è rappresentata una rete neurale A che riceve in input un valore x_t e genera in output un valore h_t. Un ciclo permette all'informazione di passare da ogni step della rete al successivo. In altre parole possiamo immaginare una *RNN* come molteplici copie della stessa rete, ognuna delle quali passa un messaggio alla rete successiva.

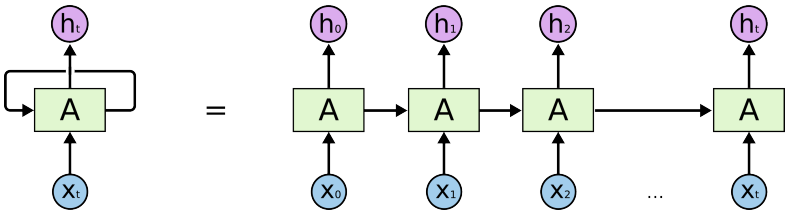

Figura 1.2 Esempio di RNN *unfolded*, cioè senza cicli

In sostanza una *RNN* riceve come input un vettore x e genera come output un vettore y ma il cuore della rete è dato dal fatto che l'output viene influenzato non solo dall'ultimo input acquisito ma anche da tutti gli input passati. Se volessimo esprimere questo concetto sotto forma di codice avremo una definizione di *RNN* che è la seguente:

```
rnn = RNN()
y = rnn.step(x)
```

La *RNN* possiede uno stato interno che viene aggiornato ad ogni esecuzione della procedura *step*. Per esempio una funzione di *step* potrebbe essere implementata come segue:

```
class RNN:
  # ...
  def step(self, x):
    # aggiornamento dello stato interno
    self.h = np.tanh(np.dot(self.W_hh, self.h)
           + np.dot(self.W_xh, x))
    # calcolo del vettore di output
    y = np.dot(self.W_hy, self.h)
    return y
```

Si può notare come la *RNN* utilizzi tre matrici: *W_hh, W_xh, W_hy*, l'*hidden state self.h* è inizializzato ome un vettore di zeri e la funzione *np.tanh* fornisce come output un valore che oscilla tra $[-1, 1]$. All'interno di questa funzione notiamo due termini, uno che si basa sullo stato dell'input attuale e uno che rappresenta il valore dello stato interno precedente. I due termini vengono sommati prima di essere elaborati dalla funzione stessa. Le matrici sono inizializzate con numeri casuali e la maggior parte di tempo speso durante la fase di *training* viene utilizzato nella ricerca di valori di queste matrici in modo che la rete si comporti nella modo desiderato, cioè che restituisca in output l'y che vorremmo avere quando forniamo in input la sequenza x.

1.7.1 RNNLM

La rete neurale utilizzata per l'analisi dei dati in questa tesi è il toolkit RNNLM. L'architettura utilizzata nel toolkit, mostrata in 1.3 viene solitamente chiamata rete di Elman o semplicemente RNN. Il *layer* di input usa una rappresentazione $1 - a - N$ delle parole precedenti $w(t)$ concatenate con il precedente stato dell'hidden layer $s(t - 1)$.

I neuroni dell'*hidden layer* $s(t)$ usano una funzione sigmoidea, cioè una funzione matematica che produce una curva sigmoide come possiamo vedere nella figura 1.4.

Questo tipo di funzione viene utilizzata per introdurre una non linearità nel modello e per assicurarsi che determinati valori rimangano all'interno di

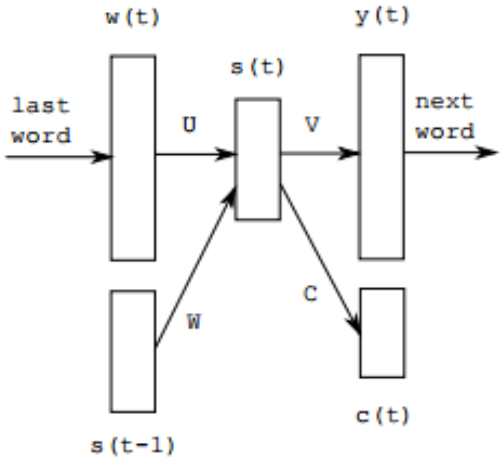

Figura 1.3 Esempio di rete neurale ricorsiva

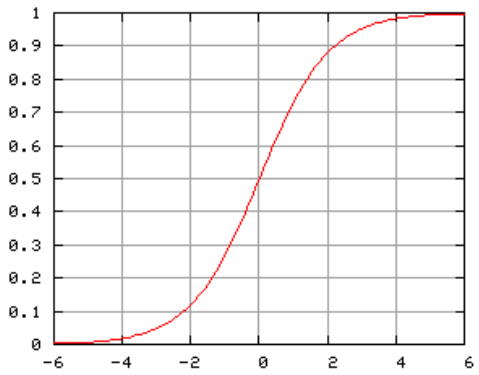

Figura 1.4 Esempio di grafico di una funzione sigmoidea

specifici intervalli. Dopo il training della rete, l'output layer $y(t)$, che ha le stesse dimensioni di $w(t)$, rappresenta la probabilità della distribuzione della parola $n+1$ partendo dalla parola n e lo stato dell'*hidden layer* nel precedente istante di tempo. Un class layer $c(t)$ può essere usato per ridurre la complessità del modello ma con un piccola diminuzione della precisione. Il training viene svolto utilizzando un *stochastic gradient descent algorithm*, cioè utilizzando una funzione di ottimizzazione locale scritta sotto forma di funzioni differenziali. La matrice W che rappresenta il *recurrent weights* (il peso degli stati passati) viene calcolata utilizzando la *backpropagation throught time*

algorithm (BPTT). Specificatamente, su RNNLM viene utilizzato un truncate BPTT, la rete cioè elaborata solo per uno specifico numero di istanti di tempo.

1.7.2 Fase di training

Nella fase di training i dati in ingresso sono attesi sotto forma di un semplice file ASCII: ogni parola deve essere separata da spazi e deve essere presente il carattere di fine linea al termine di ogni frase. Una volta specificato il corpus di input, il vocabolario viene costruito automaticamente e viene salvato insieme al file di modello generato da RNNLM. Questo significa che, nel caso si voglia limitare il vocabolario, il file di input deve essere modificato in maniera preventiva, sostituendo tutte le parole da eliminare con un token speciale (per esempio **<unk >**). Oltre al corpus, per completare la fase di training è atteso anche un file di validazioni dati per regolare il numero di training *epochs* da utilizzare e il learning rate. È anche possibile fare training di modelli senza utilizzare un file di validazione tramite l'opzione **-one-iter**.

1.7.3 Fase di test

Una volta effettuato il training il modello può essere utilizzato per valutare dati di test: l'output di questa valutazione è espresso come *perplexity* (la misura di quanto un modello probabilistico si avvicina al valore di test) e la probabilità $\log_{10}$. Dato il modello, è possibile effettuare una interpolazione lineare delle probabilità delle parole. L'input atteso da RNNLM è un file contenente una lista di frasi, ognuna anteposta con un identificatore numerico univoco, su cui effettuare lo scoring.

1.8 Word2Vec

Storicamente i sistemi di *natural language processing* trattano le parole come simboli atomici e discreti: per esempio la parola "gatto " può essere

rappresentata da **Id123** e la parola “cane ” come **Id453**. Questa codifica è arbitraria e non fornisce alcuna informazione al sistema riguardo la relazione che può esistere tra due differenti simboli. Questo significa che il nostro sistema non può utilizzare praticamente nulla di quello che ha imparato riguardo la parola “gatto ” quando sta elaborando i dati riguardanti “cane ” (per esempio il fatto che siano entrambi animali, che abbiano quattro zampe, etc.). Inoltre, rappresentare le parole sotto forma di identificatori univoci porta ad avere una base dati sparsa con la conseguente necessità di dover ottenere un maggior numero di dati per riuscire a creare un modello statistico rappresentativo; proprio questa base dati sparsa è il principale problema ad essere risolto dalla rappresentazione tramite vettori.

1.8.1 Word embedding

Con il termine *word embedding* intendiamo una rappresentazione di parole in uno spazio vettoriale continuo, ovvero un’area in cui le parole semanticamente simili sono mappate in punti vicini: questo metodo si basa sulla *distributional hypotesis*, cioè sul fatto che parole che appaiono in un determinato contesto condividono lo stesso significato semantico.

La rappresentazione di parole in uno spazio vettoriale aumenta le performance di un task di analisi del linguaggio naturale. Una delle prime applicazioni in questo campo risale al 1986 con la ricerca di Rumelharth, Hinton and Williams. Questa idea è stata successivamente utilizzata in maniera diffusa, trovando ampia applicazione in modelli di ricognizione del parlato, di traduzione automatica, nonché di numerosi altri tasks.

Recentemente, Mikolov et al. hanno introdotto il modello skip-gram: un metodo efficiente per ottenere rappresentazioni vettoriali di parole provenienti da un grande numero di dati testuali non strutturati.

Il training, servendosi di un modello Skip-gram per la creazione di vettori di parole, non utilizza moltiplicazioni tra matrici dense, al contrario della maggioranza delle reti neurali: in questo modo diventa estremamente effi-

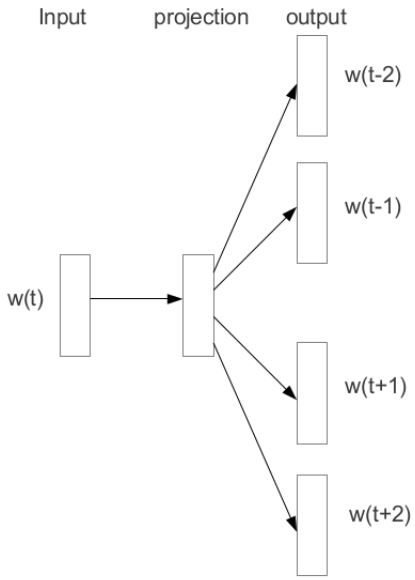

Figura 1.5modello Skipgram. L'obbiettivo è ottenere vettori di parole capaci di prevedere parole simili

ciente, in quanto una singola macchina può calcolare vettori partendo da un testo contenente più di 100 milioni di parole in meno di un giorno.

1.8.2 Modello Skip-gram

Nel modello Skip-gram l'obiettivo di training è quello di creare un vettore di parole che può essere utilizzato per prevedere vocaboli attorno ad una frase o ad un documento. Più formalmente, data una sequenza di parole $w_1, w_2, w_3, \ldots, w_T$ definita come training set, l'obbiettivo del modello è quello di aumentare la probabilità logaritmica:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_t + j|w_t) \tag{1.9}$$

dove c è la dimensione del contesto di training (che può essere una funzione basata sulla parola centrale w_t). Utilizzare un c più grande porta ad avere una precisione maggiore. In questo modo, però, si aumenta anche il tempo di calcolo. La formulazione standard del modello Skip-gram definisce $p(w_t +$

$j|w_t)$ usando la funzione softmax:

$$p(w_O|w_I) = \frac{\exp(v_{w_O} v_{w_I})}{\sum_{w=1}^{W} \exp(v_w v_{w_I})} \tag{1.10}$$

dove v_w and v'_w sono la rappresentazione vettoriale di "input" e "output" di w, mentre W è il numero di parole nel vocabolario. Questa formulazione, però, è difficilmente utilizzabile in sistemi reali perché il costo per il calcolo di $\nabla \log p(w_O|w_I)$ è proporzionale a W, che spesso è molto grande: $(10^5 - 10^7 termini)$.

1.8.3 Hierarchical Softmax

Una approssimazione del softmax computazionalmente più efficiente è il softmax gerarchico. Questo algoritmo è stato introdotto nell'analisi del linguaggio naturale da Morin and Bengio (2005) il motivo principale per cui questo algoritmo è più performante rispetto all'algoritmo classico perchè, per ottenere distribuzione di probabilità, non deve valutare tutti i W nodi di output della rete neurale, ma solo un numero $\log_2(W)$ di nodi.

Il softmax gerarchico utilizza una rappresentazione ad albero binario per il *layer* di output in cui, come foglie, troviamo le W parole e, in ogni nodo, la rappresentazione delle probabilità dei nodi figli. Questo definisce una *random walk* per assegnare la probabilità alle parole.

Più precisamente ogni parola w può essere raggiunta da un determinato percorso partendo dalla radice dell'albero. Definiamo $n(w, j)$ come il j-esimo nodo nel percorso che parte dalla radice fino a w e $L(w)$ come la lunghezza di questo percorso in modo che $n(w, 1) = radice$ e $n(w, L(w)) = w$. In aggiunta, per ogni nodo interno n, definiamo $ch(n)$ come un arbitrario nodo figlio di n e $[x]$ ha il valore 1 se x è vero e -1 altrimenti. Quindi il softmax gerarchico definisce $p(w_O|w_I)$ come segue:

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w, j+1) = ch(n(w, j))] \cdot v_{n(w,j)}) \tag{1.11}$$

dove $\sigma(x) = 1/(1+exp(-x))$. Può essere dimostrato che $\sum_{w=1}^{W} p(w|w_I) = 1$. Questo implica che il costo per calcolare $\log p(w_O|w_I)$ e $logp(w_O|w_I)$ è pro-

porzionale a $L(w_O)$ che in media è più piccolo di $\log W$. Inoltre, al contrario della formulazione stardand del softmax dove troviamo due rappresentazioni differenti $v_w ev_w$ per ogni parola w, il softmax gerarchico possiede una sola rappresentazione v_n per ogni nodo n dell'albero binario.

Il tipo di albero utilizzato ha una influenza notevole sulle prestazioni dell'algoritmo, nello specifico in word2vec è stato utilizzato un albero binario di Huffmann. In questo modo abbiamo codici di dimensioni più piccole per le parole usate con maggior frequenza, determinando un training più rapido. Inoltre è stato osservato che raggruppare le parole utilizzando la loro frequenza funziona molto bene come tecnica per aumentare la velocità di calcolo.

1.8.4 Paragraph vector

Il modello *Paragraph vector* (PV) è implementato sulla base di *word2vec* ed estende il modello di *word embedding* in modo da catturare informazioni sintattiche e semantiche provenienti da un frammento di testo (paragrafi, frasi, interi testi etc.). I PV, al contrario di altri modelli visti in precedenza, non richiedono un intervento manuale per adattarsi ai vari tipi di *task* nei quali possono venire applicati e non è nemmeno necessario utilizzare di funzioni di *weighting*: il modello è applicabile, senza ulteriori modifiche, a qualsiasi tipo di testo.

A livello implementativo, i *paragraph vector*, nascono come modifica al codice esistente di *word2vec*: entrando più nel dettaglio è stato inserito un *token*, cioè un codice fittizio, all'inizio del frammento di testo analizzato (paragrafi, frasi etc.) in questo modo, applicando gli stessi algoritmi presentati nel paragrafo 2.4.2, l'intero frammento di testo viene considerato come contesto del *token* creato. Al termine del processo di *training* il vettore di *embedding* corrispondete al *token* sarà il PV che rappresenta l'intero frammento di testo e potrà essere usato in task di classificazione.

Ogni paragrafo del documento viene rappresentato come un vettore univoco, rappresentato come una colonna della matrice D e ogni parola viene

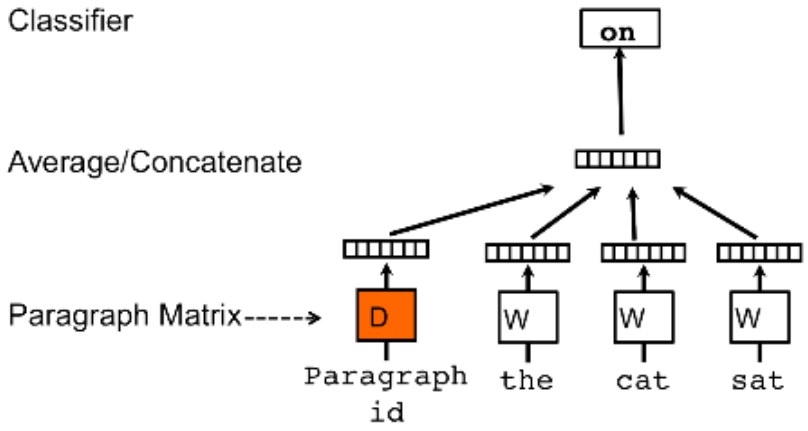

Figura 1.6 Rappresentazione del modello PV

rappresentata come una colonna nella matrice W. Il *token* del paragrafo può essere considerato come una parola aggiuntiva della frase che rappresenta l'argomento trattato nel paragrafo stesso, i PV sono condivisi per tutti i contesti generati per uno stesso paragrafo mentre i vettori di parole W sono condivisi tra tutti i paragrafi: per esempio il vettore della parola "powerful" è lo stesso in ogni parte del documento. Così come avveniva per la versione standard di *word2vec* i PV e i vettori di parole vengono generati utilizzando un algoritmo di *stochastic gradient descent* dove il gradiente viene calcolato tramite *backpropagation*. Ad ogni passo dello *stochastic gradient descent* viene considerato il contesto, di lunghezza prestabilita, di un paragrafo scelto in maniera casuale e in questo modo viene aggiornato il gradiente della rete che, infine, viene utilizzato per aggiornare i parametri del modello stesso. Se il testo è composto da N paragrafi, il vocabolario è composto da M parole, i vettori dei paragrafi hanno dimensione p e quelli di parole q allora il nostro modello utilizza $N \cdot p + M \cdot q$ parametri.

Il vantaggio principale dei PV rispetto ad altri modelli trattati, come per esempio il *BOW*, è data dalla modellizzazione della semantica delle parole: in particolare il modello riconosce come la parola "powerful" rappresenti un contetto più vicino a "strong" rispetto che all'informazione espressa dalla parola "paris". Un altro vantaggio dei *paragraph vectors* riguarda la gestione

dell'ordinamento delle parole in riferimento ad un determinato contesto: in particolare possiamo comparare questo tipo di ordinamento a quello ottenuto tramite un modello ad n-grammi.

1.9 Classificatori lineari

La risoluzione di problemi di classificazione su una mole molto grande di dati è uno dei problemi principali incontrati in applicazioni quali la classificazione di testi.

Un classificatore lineare, nel campo dell'apprendimento automatico, è diventato una delle tecnologie più promettenti e più utilizzate: l'obiettivo della classificazione lineare è quella di usare le caratteristiche degli oggetti (*Features*) per identificare a quale classe (o gruppo) appartengono. Una classificazione lineare raggiunge questo scopo effettuando una decisione sulla classificazione basata sul valore di una combinazione lineare di caratteristiche. Le caratteristiche di un oggetto sono anche conosciute come *features* e nel calcolatore sono rappresentati solitamente come un vettore chiamato vettore delle caratteristiche.

Questi classificatori funzionano molto bene per problemi pratici quali la classificazione di documenti e più in generale per problemi che utilizzano un numero molto elevato di *features*

Se definiamo un vettore di *features* con l'identificatore $\vec{x}$ allora il punteggio di output è definito come segue:

$$y = f(\vec{w} \cdot \vec{x} = f(\sum_j w_j x_j) \tag{1.12}$$

dove $\vec{w}$ è un vettore reale pesato (*weight vector*) e f è una funzione che converte il prodotto scalare di due vettori nell'input desiderato. Il *weight vector* viene generato partendo da un inseme di esempi utilizzati come training. Solitamente la funzione f è una semplice funzione che effettua il mapping di tutti i valori sopra una certa soglia nella prima classe e gli altri valori

nella seconda classe. Una versione più complessa di f potrebbe fornire una probabilità con cui un particolare oggetto appartiene ad una certa classe.

L'utilizzo dei classificatori lineari avviene spesso in situazioni in cui la velocità di classificazione è rilevante, specialmente quando $x^{\rightarrow}$ è sparso.

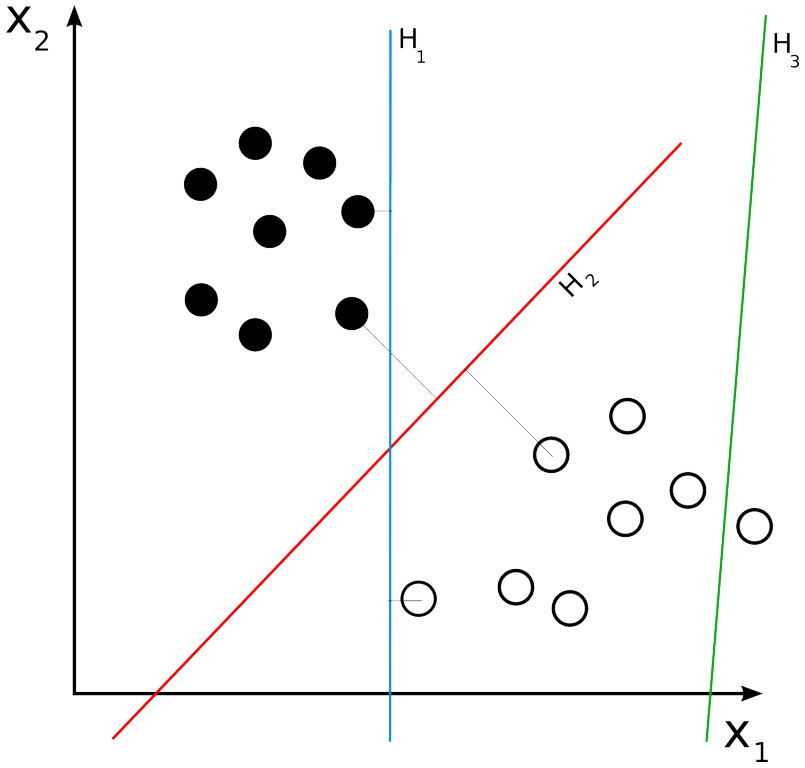

Figura 1.7 Nella figura vediamo come il classificatore H_1 e H_2 classifichino correttamente i punti pieni e i punti vuoti. Potremmo dire, inoltre, che h_2 è un classificatore migliore perché la distanza media tra i vari elementi è costante

1.9.1 Liblinear

Liblinear è una libreria opensource che implementa algoritmi di classificazione lineare. Il motivo per cui è stata scelta come strumento per questa tesi è principalmente dovuto alla sua estesa documentazione, alla semplicità d'uso, al suo già ampio utilizzo in ricerche simili e infine per la sua velocità nel calcolo dei risultati. In particolare l'ultimo punto si è rivelato fonda-

mentale per ottenere risultati in tempo ragionevole: in un confronto con un risolutore *SVM* (come *LIBSVM*) i tempi di calcolo vengono ridotti di tre ordini di grandezza (da svariate ore a pochi secondi).

Liblinear supporta due popolari classificatori binari *LR, L2-regularized logistic regression* e *SVM L2-loss linear support vector machine*, dato un inseme di coppie indicizzate $(x_i, y_i), i = 1, \ldots, l, x_i \in R^n, y_i \in \{-1, +1\}$ entrambi i metodi risolvono lo stesso problema di ottimizzazione senza vincoli ma con differenti *loss functions* $\xi(w; x_i, y_i)$:

$$min_w \frac{1}{2} w^T w + C \sum_i = 1^l \xi(w; x_i; y_i) \tag{1.13}$$

dove $C > 0$ è il parametro di penalità. Per *SVM* le due *loss functions* sono $max(1 - y_i w^T x_i, 0)$ e $max(1 - y_i w^T x_i, 0)$. Per *LR* invece abbiamo $\log(1 + e^{-y_i w^T x_i}$

Per quanto riguarda l'utilizzo, il toolkit di Liblinear fornisce due diversi eseguibili: *train* e *predict*. L'eseguibile *train* viene utilizzato per la creazione del modello e acquisisce, come input, un file ASCII suddiviso per righe. In particolare, per questa tesi, è stato usato come algoritmo per il training del modello ottenuto tramite *logistic regression*. Utilizzando la *regressione logistica* è possibile utilizzare l'eseguibile *predict* in modo che fornisca in output le stima delle probabilità per ogni riga.

Capitolo 2

Analisi dei dati

La finalità che questo capitolo si propone è quella di descrivere, inizialmente, la raccolta dei dati, la loro normalizzazione e la loro struttura. Verranno poi presentate le tecniche utilizzate per ottenere i risultati, analizzando dapprima quelli ottenuti con *RNNLM* e, a seguire, quelli raggiunti con *word2vec* e *Liblinear*. In conclusione verranno mostrati i risultati finali, ottenuti applicando le suddette tecniche.

2.1 Raccolta dati

Il primo Gennaio 2015 è iniziata la raccolta dati, provenienti da dodici canali Twitch, scelti in modo tale da rappresentare un parte omogenea di informazioni: in particolare sono stati scelti i canali dei quattro giochi più popolari con maggiore audience (valutata in base al numero medio di spettatori collegati), secondo le statistiche del mese di Gennaio 2015, della piattaforma Twitch.tv.

È opportuno specificare che i canali, e quindi anche i dati raccolti, sono tutti in lingua inglese. Questa scelta è stata determinata dalla differenza di popolarità, che emerge dal confronto tra i canali in lingua inglese e quelli in lingua italiana: nel mese di Gennaio 2015, infatti, un canale di lingua inglese

ha registrato una media di spettatori pari a trentamila persone, contro un centinaio di presenze circa rilevato in un canale italiano.

La chat presente sulla piattaforma Twitch utilizza il protocollo IRC, per cui Twitch gestisce un server dedicato [1]. Per gli utenti, l'accesso al server di chat è disponibile tramite interfaccia web, direttamente dal sito ufficiale. Per eventuali servizi di *backend* e integrazione, sono disponibili delle API ufficiali che offrono la possibilità di monitorare e interagire in tempo reale con i vari canali, previa registrazione.

Nonostante la piattaforma Twitch fornisca un archivio, contenente tutte le trasmissioni diffuse dai suoi canali, accessibile via browser, non è tuttavia possibile accedere ai dati storici delle chat. Per questo motivo è stato necessario l'utilizzo di un bot IRC che monitorasse in modo costante i canali e che provvedesse alla registrazione dei messaggi.

Per la realizzazione del bot è stato utilizzato come base il software open-source *Pierc*, esteso in modo da utilizzare il login e la registrazione al server IRC tramite API Twitch. I canali scelti per il monitoraggio sono stati gestiti tramite file di configurazione testuale e tutti i messaggi registrati sono stati salvati in un database *MySQL*. La struttura del database è mostrata nella tabella 2.1.

Campo	**Tipo**	**Descrizione**
id	int	Id incrementale
channel	varchar	Nome canale
name	varchar	Nome utente
time	datetime	timestamp
message	text	messaggio inserito
type	varchar	tipo di messaggio IRC

Tabella 2.1 Struttura del database

Analizzando la struttura della tabella, è opportuno notare come si sia

[1] `irc://irc.twitch.tv`

scelto di aggiungere, oltre al testo del messaggio, anche il *timestamp* e il canale di appartenenza. Questi dati hanno permesso di affinare la ricerca e di effettuare analisi specifiche, prendendo in considerazione solo particolari giornate oppure dati provenienti da un solo, giocatore.

2.2 Normalizzazione del corpus

Uno dei principali problemi incontrati durante l'analisi dei dati è consistito nella normalizzazione del corpus. È importante sottolineare che un'indagine su questo tipo di dati non era mai stata affrontata prima, per cui non è stato possibile trovare una metodologia che fosse già stata applicata a questo tipo di studio.

Si evidenzia, inoltre, che Twitch, pur essendo una piattaforma relativamente giovane, sviluppatasi particolarmente negli ultimi tre anni, è cresciuta molto velocemente, creando una sottocultura di Internet con frasi ricorrenti, giochi di parole ed espressioni linguistiche nuove. Questo gergo si è talmente esteso da influenzare addirittura altre piattaforme, come ad esempio lo *streaming* su Youtube, portando alla formazione di particolari tipologie di messaggi, suddivise secondo le modalità indicate dalla tabella 2.2, attraverso un'analisi manuale.

Messaggio	Interpretazione
¯_(ツ)_/¯	emoticons UTF8
˚Д˚ GO TEAM ˚Д˚	Frasi contenenti codici UTF8
S U F F E R B O Y S	enfasi
	emoticons multiple

Tabella 2.2 Differenti tipi di messaggio su Twitch

Fortunatamente, oltre a questi messaggi speciali, su Twitch è presente anche una grande quantità di messaggi standard, come possiamo vedere nell'esempio 2.2.1.

Esempio 2.2.1. Esempio di alcuni messaggi di chat:
 800 viewers 


Trump are you a proplayer? wait dont answer
I always wait until my opponent has 7 minions so i can get max unleash the hounds value 
Play wolf among is 
if you spam your abilities fast enought with this cham , you can remix sandstorm

Per normalizzare il corpus è stato usato *Twokenizer*, un tokenizzatore sviluppato dalla Carnagie Mellon University per gestire dati provenienti da Twitter. La scelta di utilizzare questo tokenizzatore è dovuta al fatto che, a parte i casi particolari indicati nella tabella 2.2, l'unica differenza rilevante tra i tra i messaggi di Twitch e quelli di Twitter è costituita dagli *hashtag*, utilizzati solo in Twitter.

Twokenizer è stato opportunatamente modificato, implementando regole per la sostituzione di tutte le emoticon espresse in caratteri UTF8 con un corrispettivo *token*, attraverso una tabella di mapping, una parte della quale viene mostrata nella figura 2.1). In questo modo si è evitato sia di dover gestire emoticons composte da caratteri multipli, sia di gestire i caratteri UTF8 non supportati da *word2vec*, come vedremo in seguito. In secondo luogo si è provveduto alla sostituzione di tutti gli *URL* con il token **URL**; sono stati ignorati, infine, i simboli # (che indicano i cosiddetti hashtag, non presenti su Twitch), considerandoli come normale segno di punteggiatura. Come ultimo passo si è scelto di modificare le frasi che esprimono enfasi, scritte in lettere maiuscole tutte separate da uno spazio, eliminando gli spazi e unendo la parola. Utilizzando sempre l'esempio della tabella 2.2 la frase "S U F F E R B O Y S" è stata trasformata in "SUFFERBOYS".

Vista la mole dei dati raccolti, che ammontano a quasi sei milioni di messaggi, è stato necessario modificare anche il metodo di esecuzione del

¬_¬	<emo1>
¯_(⊙﹏⊙)_/¯	<emo2>
¯_(ツ)_/¯	<emo3>
°ω°	<emo4>
°Д°	<emo5>
°‿‿°	<emo6>
°ฺ°	<emo7>
٩◔◔۶	<emo8>
٩(×̯×)۶	<emo9>
٩(ಥ_ಥ)۶	<emo10>
٩(•̮•)۶	<emo11>
٩(●̮•)۶	<emo12>
٩(●̮●)۶	<emo13>
٩(｡•̀‿•｡)۶	<emo14>

Figura 2.1 Parte di tabella di mapping per le emoticons UTF8

tokenizzatore, che nella sua versione standard, è stato sviluppato per un'elaborazione *single thread*. È stato scritto un semplice *wrapper* in modo da effettuare un *preprocessing* del corpus, suddividendolo in n parti distinte, dove con n è indicato il numero di processori presenti sulla macchina ed eseguendo su ciascuna di esse *Twokenizer* in maniera concorrente. Il wrapper, infine, ha effettuato l'unione di tutti i dati ottenuti mantenendo lo stesso ordinamento del file iniziale.

2.3 Analisi Emoticons

Particolare attenzione è stata posta nell'analisi delle emoticon di *Twitch* che, al contrario di quelle viste nel paragrafo precedente, sono costituite da normali parole (Es: *kappa, 4head, biblethump*), queste parole sono trattate in maniera speciale da *Twitch* che le renderizza facendole apparire sotto forma di immagine. Per esempio la parola *kappa* viene sostituita con  , *4head* con  e *biblethump* con 

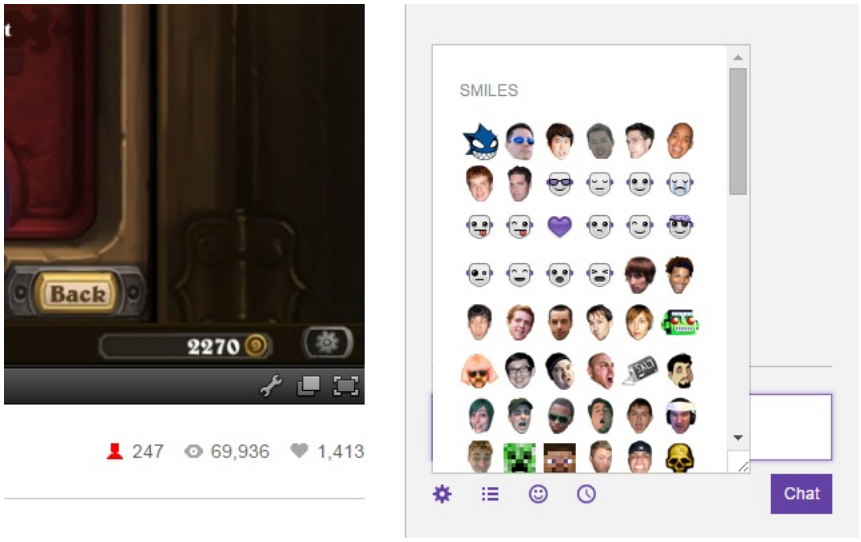

Figura 2.2 Screenshot dell'interfaccia browser di Twitch che mostra le emoticons renderizzate come appaiono all'utente finale.

Le emoticons fornite da twitch sono centoventi, esaminando tutto il corpus raccolto, si è scelto di effettuare un'analisi manuale limitandola alle sole venti emoticon usate con maggior frequenza. Questa scelta è stata determinata dal riscontro fornito dalle percentuali di utilizzo: le venti emoticon selezionate rappresentano il 93% di tutte le emoticon mentre le restanti cento coprono soltanto il 7%. Analizzando alcune centinaia di messaggi per ognuna di queste emoticon, è stata fatta una classificazione della tipologia di frasi in cui l'emoticon è utilizzata. Il risultato è quello mostrato nella tabella 2.3.

Nome Emoticon	Rappresentazione	Significato	Polarity
4Head		approvazione	positiva
babyrage		insofferenza	negativa
biblethump		disappunto	negativa
brokeback		ironia	positiva
dansgame		critica	negativa
elegiggle		inronia	positiva
pogchamp		spam	non considerato
residentsleeper		noia	negativa
swiftrage		rabbia	negativa
wutface		spam	non considerato
failfish		disappunto	negativa
frankerz		spam	non considerato
heyguys		annunci e saluti	positiva
kappa		ironia	positiva
kappapride		ironia	positiva
kreygasm		battute	positiva
mrdestructoid		viewbotting	non considerato
notlikethis		offese	negativa
osrob		spam	non considerato
pjsalt		disappunto	negativa

Tabella 2.3 Analisi delle emoticons

Analizzando la tabella è facile notare come sia possibile dividere le emoticons in tre classi separate:

- **Positiva**: rappresenta frasi ironiche, battute e frasi che esprimono un apprezzamento sul contenuto del canale.
- **Negativa**: rappresenta un giudizio negativo sul contenuto del canale o sugli argomenti trattati in chat.

- **Non considerata**: in questa classe troviamo le emoticon che non esprimono alcun giudizio e vengono quasi sempre utilizzate solo per creare "confusione " nella chat: emoticon come *mrdestructdroid*  o *frankerz*  , ad esempio, vengono utilizzate in frasi senza significato, contenenti quasi sempre solo la stessa emoticon ripetuta, in modo da generare un'enorme quantità di messaggi in chat. Questo comportamento viene utilizzato soltanto per impedire una discussione in chat e per creare disordine, come avviene talvolta anche nei forum: possiamo, quindi, definirla vero e proprio *spam*. Per quanto concerne la presente tesi, è bene sottolineare che queste frasi sono state mantenute nel corpus, pur senza attribuire loro alcun tipo di polarità: sono considerate frasi neutre.

Positive	Negative	Non considerate
4Head	babyrage	wutface
anele	biblethump	pogchamp
brokeback	dansgame	frankerz
elegiggle	residentsleeper	osrob
heyguys	failfish	mrdestructoid
kappa	pjsalt	
kappapride	notlikethis	
kreygasm		

Tabella 2.4 Lista di emoticons raggruppata per polarity

2.4 Strumenti di analisi

Per sviluppare questa tesi sono stati utilizzati i vari strumenti descritti nel capitolo precedente ed in particolare **RNNLM** versione 0.4b [2], **word2vec** re-

[2]`https://goo.gl/e9AxGR`

vision 42, [3] a cui è stata applicata la patch per i *paragraph vectors* e **Liblinear** versione 2.1 [4].

Per automatizzare e integrare i processi di *test*, di *training* di analisi e di integrazione tra i vari strumenti, sono stati sviluppati diversi programmi e script utilizzando svariati linguaggi: *BASH, Python* (sia la versione 2.7 che 3.3) e *C#*.

Tutto il software scritto per questa tesi, infine, è disponibile in un *repository Git*. [5]

2.4.1 RNNLM

Dal corpus iniziale sono state estratte tutte le frasi contenenti le quindici *emoticon* individuate in precedenza: il corpus risultante, a cui assegnamo il nome **EMOLABEL**, contiene solo i messaggi in cui è presente almeno una *emoticon*.

I messaggi contenuti in **EMOLABEL** sono stati categorizzati utilizzando come elemento discriminante le *emoticon* individuate nella tabella 2.4, in modo da realizzare due corpus distinti, il primo dei quali contenente tutti i messaggi dove è presente almeno una *emoticon* positiva ed il secondo comprensivo di quelli contraddistinti da *emoticon* negative. Sulla base della metodologia presentata da (Youngjoong and Jungyun, 2010; Pak and Paroubek, 2013; Turney, 2012; Read and Carrol, 2013; Roberts et al., 2010) si è quindi proceduto ad un primo esperimento per la cui esecuzione sono state utilizzate, come corpus da cui estrarre i messaggi necessari per fare train e test, le frasi contenute in **EMOLABEL**: successivamente tutte le emoticon usate per compiere la classificazione sono state rimosse. Al corpus creato tramite questo processo è stato assegnato il nome **NOEMO**.

Il risultato che ci si è prefissi di raggiungere con questo tipo di approccio è quello di capire se le le informazioni sulla polarità siano contenute nelle

[3] `http://goo.gl/IWUQoE`
[4] `http://goo.gl/4daeR3`
[5] `https://github.com/ManofWax`

parole e nella struttura del testo restante, oppure si trovino soltanto nelle emoticon che abbiamo eliminato. È quindi molto importante tenere presente che i risultati illustrati in questo paragrafo non ci forniscono alcuna informazione riguardo alla bontà del modello implementato: i risultati finali, infatti, verranno presentati utilizzando un'altra metodologia, come descritto dettagliatamente nel paragrafo 2.5.1

Seguendo le ricerche di Mikolov et al. (2011b) e utilizzando il corpus **NOEMO** appena creato sono stati generati due file differenti, il primo dei quali è stato realizzato partendo da tutte le frasi che contenevano *emoticon* positive e il secondo comprendente tutte le frasi in cui erano contenute *emoticon* negative: in questi file è stato mantenuto lo stesso ordinamento delle frasi originali e da entrambi sono state escluse tutte le frasi al cui interno fossero contenute, contemporaneamente, sia *emoticon* positive che *emoticon* negative.

Dopo aver effettuato questa prima suddivisione, si è proceduto ad un'ulteriore distinzione in quelli che possiamo definire come file di *train* e di *test*. I file di *train* utilizzati sono composti da ottantamila frasi selezionate in maniera casuale tra tutte quelle presenti nel file originale; i file di test, invece, sono composti da ventimila frasi, anch'esse scelte casualmente tra quelle del file originale. Particolare attenzione è stata posta nell'evitare che i messaggi selezionati per il file di *train* venissero utilizzate anche in quello di *test*.

Il modello, per ogni file, è stato generato utilizzando la seguente configurazione:

```
head -n 75500 $dir > $dir.train
tail -n 500 $dir > $dir.valid

./rnnlm -rnnlm $dir.model -train $dir.train -valid $dir.valid \
        -hidden 50 -direct-order 3 -direct 200 -class 100 -bptt 4 \
        -bptt-block 10 -binary
```

Successivamente sono stati utilizzati i modelli generati dai due file di *train*, quello contenente frasi positive e quello contenente frasi negative con i

due file di *test*. I file di *test* sono stati uniti in modo da avere un file di test unico e utilizzando la funzione **n-best scoring** di RNNLM su cui vengono generati due file risultati differenti, uno basato sul modello positivo e uno sul modello negativo.

```
awk 'BEGIN{a=0;}{print a " " $0; a++;}' < test.txt > test-id.txt
./rnnlm -rnnlm model-pos -test test-id.txt -nbest > pos-score
./rnnlm -rnnlm model-neg -test test-id.txt -nbest > neg-score
```

Vengono infine uniti e valutato il file contenete i risultati: se il rapporto tra il risultato ottenuto dal modello negativo e quello raggiunto dal modello positivo è maggiore di 1, allora alla frase viene attribuito un valore positivo, in caso contrario le viene assegnato un valore negativo. Sono stati eseguiti cinque test, scegliendo ogni volta in maniera casuale le frasi utilizzate per il file di *train* e per quello di *test*. I valori ottenuti da una prima analisi sono i seguenti:

	Accuracy	Precision	Recall	F-Score
Media 5 run	78.285	0,783	0,782	0,782

Tabella 2.5 Risultato RNNLM

		Classificati come:	
		Positivi:	Negativi:
Reali	Positivi:	15941	4059
	Negativi	4045	15955

Tabella 2.6 Confusion matrix RNNLM

Dai risultati ottenuti si può rilevare come il modello riesca ad estrarre informazione di polarità dalle parole nelle frasi e come riesca a discriminare correttamente le due classi costruite partendo dalle emoticon. Si può dedurre,

quindi, che l'utilizzo di emoticon come *label* per esprimere la polarità delle frasi è un buon sistema di categorizzazione.

2.4.2 Word2Vec

Seguendo il medesimo procedimento usato per effettuare i test con *RNNLM* sono stati utilizzati i file di *train* e di *test* per svolgere gli stessi test di *accuracy* tra messaggi provenienti da due differenti emoticon. Anche in questo caso è stato utilizzato il corpus **NOEMO**, in questo modo è possibile effettuare un'analisi sulle informazioni di polarità fornite dalle restanti parole dei messaggi analizzati. Ancora una volta, quindi, l'interpretazione dei risultati deve essere svolta in maniera analoga a quella usata per analizzare le risultanze presentate nel paragrafo 2.4.1, verificando, cioè, se l'analisi tramite *word2vec* consenta di estrarre informazioni anche dal resto delle frasi.

È importante sottolineare che, nonostante venga utilizzato un sottoinsieme dei dati per effettuare le prove, i vettori sono stati generati ricorrendo all'intero corpus e sono stati generati utilizzando il seguente codice:

```
./word2vec -train vec-id.txt -output vectors.txt -cbow 0 \
           -size 100 -window 10 -negative 5 \
           -hs 0 -sample 1e-4 -threads 40 -binary 0 \
           -iter 20 -min-count 1 -sentence-vectors 1
```

Sono state effettuate alcune prove, sia cambiando la lunghezza dei vettori, (utilizzandone da 100, 200 e 300 elementi), sia ricorrendo all'algoritmo cbow che skip-gram, senza che tuttavia emergessero particolari differenze nei risultati: per questo motivo si è scelto di utilizzare vettori da 100 elementi e l'algoritmo skip-gram.

Per quanto riguarda la valutazione del risultato è stato utilizzato *Liblinear*, ed in particolare è stato usato l'algoritmo di *logistic regression* per calcolare l'*accuracy* sul file di *test*:

```
./train -s 0 train.txt model.logreg
./predict -b 1 test.txt model.logreg out.logreg
```

Infine, utilizzando lo stesso set di dati descritto nel del paragrafo 2.4.1, è stata calcolata l'*accuracy* tra file positivi e negativi:

	Accuracy	Precision	Recall	F-Score
Media 5 run	76.285	0,763	0,762	0,762

Tabella 2.7 Risultati word2vec

		Classificati come:	
		Positivi:	Negativi:
Reali	Positivi:	15141	4859
	Negativi	4745	15255

Tabella 2.8 Confusion matrix word2vec

Da questi risultati si può affermare che anche *word2vec* costituisce un valido modello per effetuare una analisi adeguata, e soprattuto mostra come le parole utilizzate possono fornire una informazione sulla *polarity*.

Come dimostra uno studio effettuato utilizzando l'algoritmo *t-SNE*, cioè un algoritmo di *Distributed Stochastic Neighbor Embedding* , è stato possibile ridurre le dimensioni dei vettori generati tramite *word2vec* e realizzare un grafico cartesiano rappresentante la "distanza " tra le varie emoticon. In particolare, in questo esperimento, sono stati generati vettori di parole senza utilizzare i *paragraph vecotrs*, nello specifico è stato utilizzato il seguente comando:

```
./word2vec -train text8 -output vectors.bin -cbow 1 \
-size 200 -window 8 -negative 25 -hs 0 \
-sample 1e-4 -threads 20 -binary 1 -iter 15
```

Una volta costruiti i vettori, con l'aiuto di uno script *Python* e *gensim*, una libreria che permette l'integrazione e la manipolazione dei dati prodotti

da *word2vec* in direttamente tramite interprete *Python*, sono stati estratti i vettori relativi alle centoventi *emoticon* considerate e tramite l'algoritmo *t-SNE* è stata possibile costruire la figura 2.3.

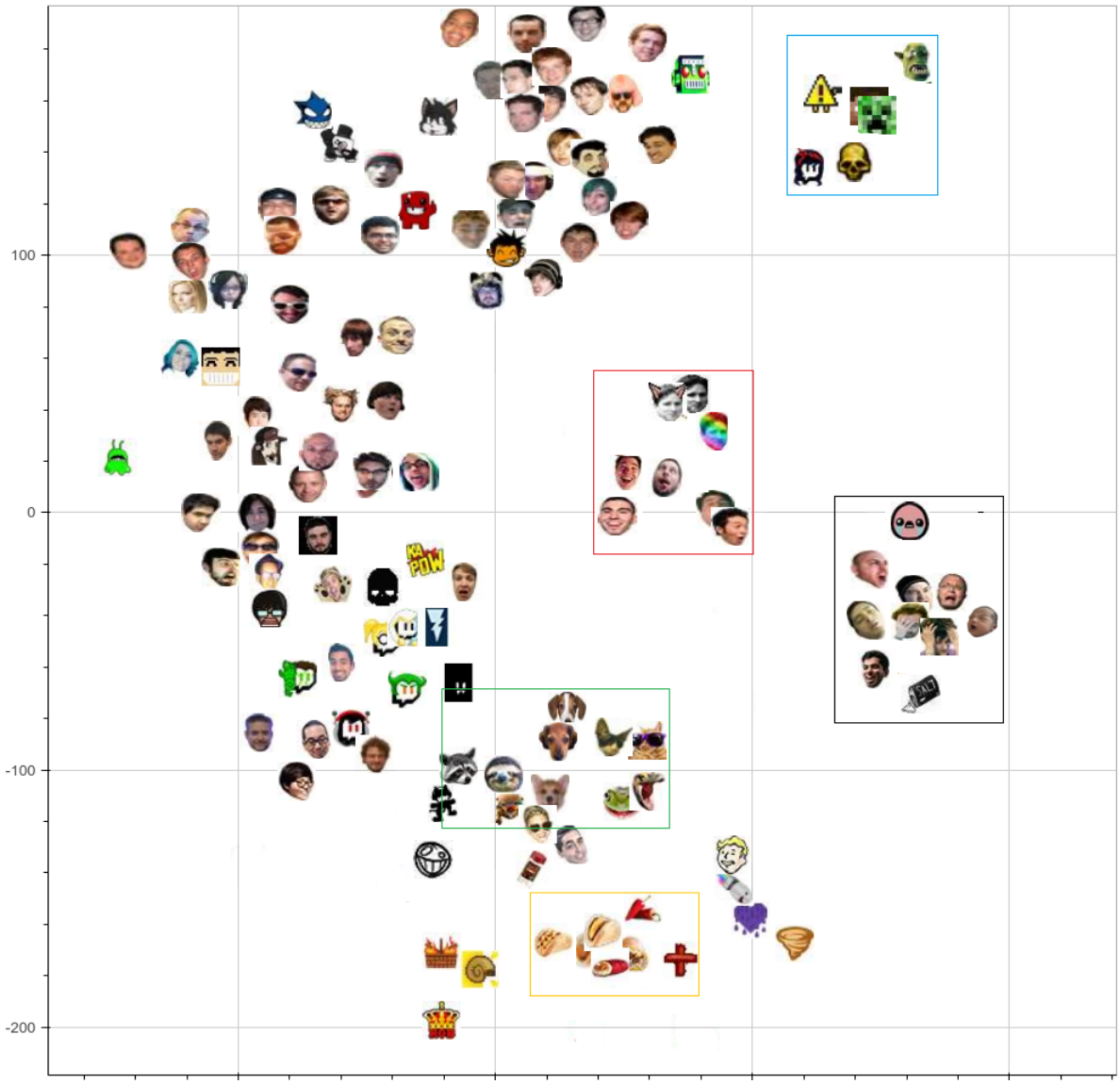

Figura 2.3 Grafico rappresentante i vettori di emoticon.

Dal grafico sopra rappresentato è possibile notare come, effettivamente, emoticon simili fra loro vengano raggruppate: possiamo notare come le emoti-

con trattate in questa tesi compongano due differenti insiemi: quelle negative sono raggruppate nel centro-destra, indicate con un rettangolo nero, mentre quelle positive formano un'insieme centrale indicato con un rettangolo rosso. Sono stati evidenziati anche tre ulteriori gruppi: in basso, evidenziato in giallo, vediamo raggruppate le emoticon rappresentati cibi e bevande, un pò più in alto, evidenziate in verde, sono presenti emoticon con sembianze di animale. Infine, raggruppate con il colore azzurro, vediamo in alto a destra emoticon riguardanti uno specifico gioco, particolarmente discusso nella piattaforma Twitch.

2.5 Analisi utilizzando entrambi i modelli

Prima di effettuare i test conclusivi, si è scelto di provare ad effetuare un esperimento applicando dapprima *RNNLM* e successivamente *word2vec e Liblinear* sullo stesso set di dati, in particolare il corpus **NOEMO**.

Come suggerito dalla ricerca di Le and Mikolov (2014), per confrontare i risultati ottenuti dall'utilizzo di entrambi i modelli sono state impiegate le probabilità logaritmiche precedentemente generate usando l'interpolazione lineare. Più formalmente, la probabilità complessiva è stata definita come la media geometrica pesata dei modelli:

$$p(y = +1|x) = \prod p^k(y = +1|x)^{\alpha_k}, con \alpha_k > 0 \tag{2.1}$$

La ricerca di α è stata effettuata utilizzando un *brute force grid search*, quantificando i valori dei coefficienti nell'intervallo $[0, 1]$ in incrementi di 0.1. Questa ricerca viene controllata tramite un corpus di validazione: nel particolare è stato usato un *5-fold cross validation*. Nel dettaglio è stato suddiviso il dataset di *test*in cinque parti di uguale numerosità e, ad ogni passo, la parte (1/5)-esima del dataset è stata usata come *validation dataset*, mentre la restante parte costituisce il training dataset in questo modo sono stati evitati i problemi legati all'*overfitting*.

2.5.1 Risultati

Anche questo esperimento è analogo a quelli presentati in precedenza: in particolare, per realizzare i test, è stato generato un corpus di training, partendo da **NOEMO**, di ottantamila messaggi per file (file positivo e negativo), scelte casualmente tra tutto il corpus, e un test, partendo sempre da **NOEMO** composto da ventimila messaggi per file. I vettori sono stati generati utilizzando l'intero corpus, non limitandosi quindi al numero di righe prese in considerazione per i file di training e di test. Anche in questo caso i dati di training e di test sono ottenuti dalla categorizzazione iniziale effettuata tramite *emoticon*, questa analisi è stata usata per la ricerca del coefficiente α prima di applicare il modello con i dati annotati manualmente.

Nella seguente tabella sono riportati i risultati ottenuti:

	Accuracy	Precision	Recall	F-Score
Media 5 run	81.165	0,816	0,812	0,814

Tabella 2.9 Risultati RNNLM + word2vec

		Classificati come:	
		Positivi:	Negativi:
Reali	Positivi:	16575	3425
	Negativi	3411	16589

Tabella 2.10 Confusion matrix RNNLM + word2vec

Nell'analizzare i dati ottenuti si è proceduto ad eseguire anche un'ulteriore prova: è stato effettuato un test rimuovendo le ripetizioni di frasi o di parole all'interno dello stesso messaggio. Si è notato, ad esempio, che per esprimere enfasi gli utenti tendono a ripetere più volte il messaggio, di modo che si possano trovare trovare frasi del tipo "Good Game Good Game Good Game Good Game ". Utilizzando sempre la stessa metodologia

applicata finora, si è proceduto ad effettuare una prova, che è consistita nel rimuovere le ripetizioni: la frase tipo, quindi, è stata ridotta a “Good Game  ”.

Tuttavia, come si può vedere dai risultati nella tabella 2.11, questo meccanismo non è stato applicato ai test successivi, in quanto è emerso che lo stesso tende ad alterare il corpus originale e a offrire valori di precisione inferiori a quelli ottenuti.

	Accuracy	Precision	Recall	F-Score
Media 5 run	79.165	0,796	0,792	0,794

Tabella 2.11 Risultati rimuovendo ripetizioni

		Classificati come:	
		Positivi:	Negativi:
Reali	Positivi:	19411	589
	Negativi	680	19320

Tabella 2.12 Confusion matrix rimuovendo ripetizioni

2.5.2 Analisi su corpus annotato manualmente

I test per la valutazione del modello utilizzato sono stati svolti facendo ricorso ad un corpus, che chiameremo **MANTEST**, composto da circa cinquemila messaggi annotate manualmente: si tratta, per la precisione, di 5631 messaggi. Di queste, 2670 sono state classificati come positive e le restanti 2871 come negative. I dati di training utilizzati sono quelli provenienti dal corpus **EMOLABEL**, quindi il corpus contenente anche tutte le *emoticon*, mentre come dati di test sono stati utilizzati tutti i messaggi presenti in **MANTEST**.

I risultati ottenuti sono riportati nelle tabelle seguenti:

	Accuracy	Precision	Recall	F-Score
Media 5 run	76.95	0,799	0,757	0,770

Tabella 2.13 Risultati testo annotato manualmente

		Classificati come:	
		Positivi:	Negativi:
Reali	Positivi:	2110	560
	Negativi	697	2174

Tabella 2.14 Confusion matrix testo annotato manualmente

Come baseline è stata calcolata la polarità, sugli stessi dati usati nel test precedente, tramite l'utilizzo di un modello *BOW*, in particolare utilizzando l'implementazione fornita dal *framework* di *machine learning* Weka e come categorizzatore *Liblinear* con gli stessi parametri usati per l'analisi effettuata con *word2vec*. I risultati sono indicati nella tabella 2.15

	Accuracy	Precision	Recall	F-Score
Media 5 run	73,18	0,751	0,716	0,736

Tabella 2.15 Risultati BOW

		Classificati come:	
		Positivi:	Negativi:
Reali	Positivi:	2007	663
	Negativi	799	2072

Tabella 2.16 Confusion matrix BOW

Questi risultati ci mostrano come il modello attuale possa essere utilizzato per compiere un'analisi su dati reali e come il risultato ottenuto sia maggiormente attendibile rispetto a quello che si otterrebbe ricorrendo all'applicazione di un'analisi basata su BOW e Liblinear. Inoltre questi risultati mostrano come un approccio totalmente non supervisionato sia molto ben applicabile nell'ambito di Twitch.tv, infatti partendo da dati di *training* categorizzati soltanto attraverso *emoticon* vediamo come il modello riesca ad ottenere valori di *accuracy* e *F-score* del $76_{\%}$.

2.6 Analisi dell'andamento giornaliero

Come caso di studio sono state analizzate le chat e le trasmissioni giornaliere prodotte da alcuni utenti del sito, contenenti una media di quindicimila messaggi ciascuno, e l'indagine si è focalizzata, in particolare, sull'indice di gradimento della trasmissione, valutata, mediante i metodi finora descritti, in base al numero di frasi positive e negative per ogni istante di tempo.

In prima analisi, partendo dai dati raccolti, è stata effettuata una suddivisione degli stessi utilizzando le informazioni registrate riguardanti i *timestamp* e il canale di appartenenza di ogni messaggio. In secondo luogo, i dati sono stati elaborati e normalizzati con l'utilizzo di Twokenizer, nello stesso modo in cui erano stati normalizzati i dati nei *test* precedenti. Infine l'intera giornata è stata suddivisa in intervalli di 5 minuti e, utilizzando i modelli creati precedentemente, è stata determinata la polarità per ogni frase; per ciascun intervallo sono state calcolate le somme di tutte le frasi positive e negative.

I risultati ottenuti sono stati rappresentati con un grafico in cui le ascisse rappresentano gli istanti di tempo, mentre le ordinate stanno ad indicare la media delle frasi positive e negative: un valore positivo indica una maggioranza di frasi positive e viceversa. Nel grafico sono presenti due tipi di dati differenti: quelli indicati in azzurro sono i valori calcolati utilizzando i modelli presenti nella tesi, mentre quelli rappresentati in rosso i dati dati

ottenuti solamente dalle emoticon: contando solo quelle contenute nelle frasi e attribuendo loro il valore rappresentato nella tabella 2.3, si è stabilita la polarità, positiva o negativa, assunta dalla frase.

Gli obbiettivi di questo esperimento sono due: inanzitutto viene calcolata la correlazione tra i dati ottenuti semplicemente dalle *emoticon* ,utilizzando la stessa metodologia per la creazione del corpus **EMOLABEL**, con quelli ottenuti dal sistema implementato in questa tesi utilizzando *RNNLM* e *word2vec*. In particolare, se la correlazione è molto elevata, significa che il modello utilizzato in questa tesi non ottiene ulteriori informazioni rispetto ad un semplice calcolo basato sulla frequenza delle *emoticon*. In secondo luogo vengono analizzati i picchi presenti nel grafico, cioè i momenti dove il modello identifica una maggior polarità nei messaggi, e stabilendo la polarità manualmente di circa cento messaggi presenti in ogni picco si verifica se il modello stia effettivamente rilevando la polarità corretta.

A seguire, verranno presentatate tre casi studio, selezionati per le loro particolarità: per una maggiore chiarezza, i relativi grafici sono stati raffigurati a paigna intera a fine capitolo.

Analisi utente A

Innanzitutto, analizzando il grafico dell'utente A, notiamo come i dati calcolati con i modelli coincidano esattamente con quelli calcolati usando solamente le emoticon: il coefficiente di correlazione tra i due dati è 0.91. Ciò deriva dal fatto che la chat dell'utente A è caratterizzata da un uso massiccio di emoticons, tant'è che, nella giornata analizzata, il 94% delle frasi conteneva almeno un emoticon: poiché il nostro modello è basato su frasi etichettate con le suddette emoticons, diventa molto chiaro il motivo per cui i due dati coincidano. I dati indicati dalle due etichette $A1$ e $A2$ rappresentano particolari eventi positivi (vittorie del giocatore), entrambi gli eventi sono state selezionati dall'utente A in un video di *highlights* prodotto successivamente, questo mostra come il modello utilizzato possa essere sfruttato per generare una lista di punti salienti degli eventi accaduti durante la giornata.

Analisi utente B

Il discorso è differente per quanto riguarda il grafico dell'utente B, in particolar modo se consideriamo che in questo caso le emoticon presenti in chat sono in numero assai ridotto se confrontate con quelle dell'utente A: infatti meno del 50% delle frasi ne contiene una. Il coefficiente di correlazione, in questo caso, scende a -0.04, e possiamo vedere come nel grafico generato solo dalle emoticon sia presente un numero molto inferiore di picchi. Andando ad analizzare cento messaggi in maniera manuale nelle zone dei picchi presenti nella registrazione giornaliera, si è potuto constatare come nei punti indicati da $B1$, $B2$, $B3$ siano presenti eventi negativi, in particolare nei primi due, riguardanti una sessione di gioco particolarmente sfavorevole che ha portato alla sconfitta della squadra, mentre nella terza, oltretutto di durata più lunga, è stato presente un disservizio che ha portato alla riduzione della qualità video e audio della trasmissione stessa. È interessante notare, inoltre, che durante l'evento $B3$ i dati calcolati solo attraverso le emoticon, indicano un picco positivo, mentre i dati calcolati attraverso i modelli mostrano un picco ampiamente negativo. Analizzando le frasi con emoticon positive, si è stabilito che in questo caso i messaggi contenuti erano di natura sarcastica.

Analisi utente C

Esaminando, infine, il grafico generato per l'utente C, possiamo notare come, nella prima parte del grafico, ci sia un'alta correlazione tra i due dati mostrati, e cioè un valore pari quasi allo 0.60 mentre nella seconda parte la correlazione scende a -0.09. Dall'analisi del video emerge che, nella prima parte l'utente C stava mostrando un gioco particolarmente semplice, con conseguente basso livello di interazione tra gli utenti della chat che, infatti, si limitavano a commentare con frasi corte pur ricorrendo in maniera massiccia all'uso di emoticon, cosi' come accadeva per l'utente A. Nella seconda seconda parte, invece, la trasmissione passava ad un argomento più serio, distaccandosi dal contesto dei giochi e trattando temi riguardanti la politica americana: questo segmento di video mostra con grande evidenza una for-

tissima diminuzione dell'utilizzo delle emoticons, a favore di una maggiore lunghezza dei messaggi che in questo caso esprimono anche pareri e opinioni complesse. Effettuando una verifica manuale sui contenuti, si evince che, effettivamente, come rappresentato dall'andamento negativo, la maggior parte degli ascoltatori non condivideva il parere dell'utente C e quindi esprimeva commenti negativi

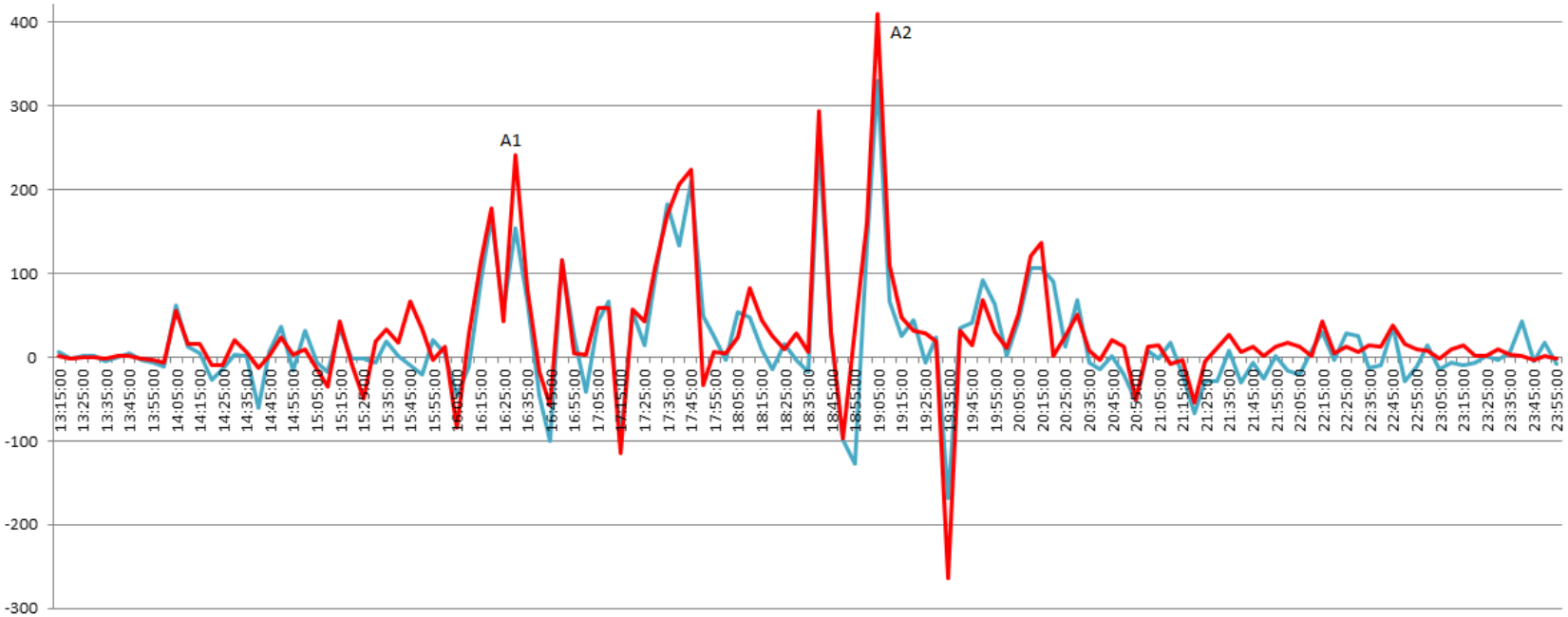

Figura 2.4 Andamento giornaliero utente A. Il grafico rosso rappresenta la polarità dei messaggi calcolata utilizzando solo le *emoticon* (come effettuato per la creazione del corpus **EMOLABEL**). Il grafico blu rappresenta la polarità calcolata con *RNNLM e word2vec*. L'altissima percentuale di *emoticon* nei messaggi fa si che la correlazione tra i dati calcolati dai due sistemi sia molto elevata. Verificando manualmente la polarità nei due picchi $A1$ e $A2$ si è stabilito che essi rappresentano particolari eventi positivi (vittorie del giocatore).

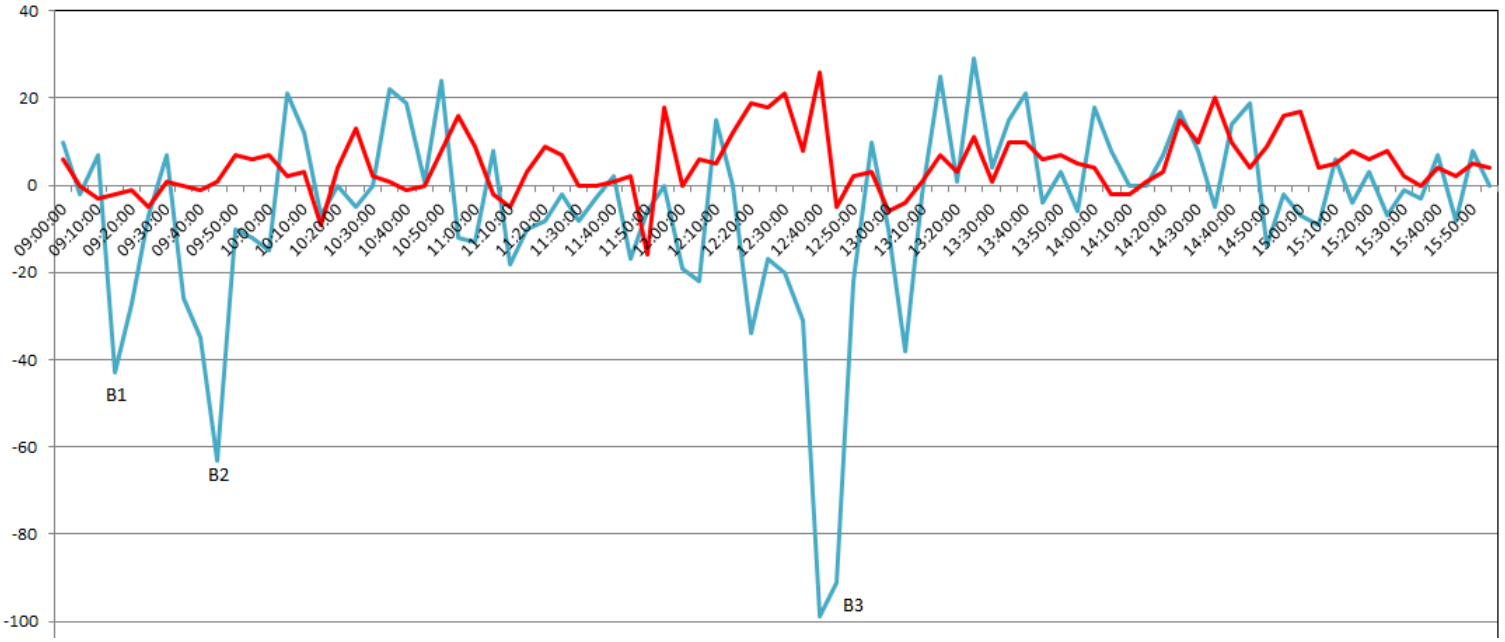

Figura 2.5 Andamento giornaliero utente B. Il grafico rosso rappresenta la polarità dei messaggi calcolata utilizzando solo le *emoticon* (come effettuato per la creazione del corpus **EMOLABEL**). Il grafico blu rappresenta la polarità calcolata con *RNNLM e word2vec*. Il basso coefficente di correlazione tra i due grafici è dovuto alla forte mancanza di *emoticon*, ed analizzando manualmente cento messaggi in corrispondenza di $B1$, $B2$ e $B3$ si è stabilito come il modello utilizzato individui correttamente una polarità negativa.

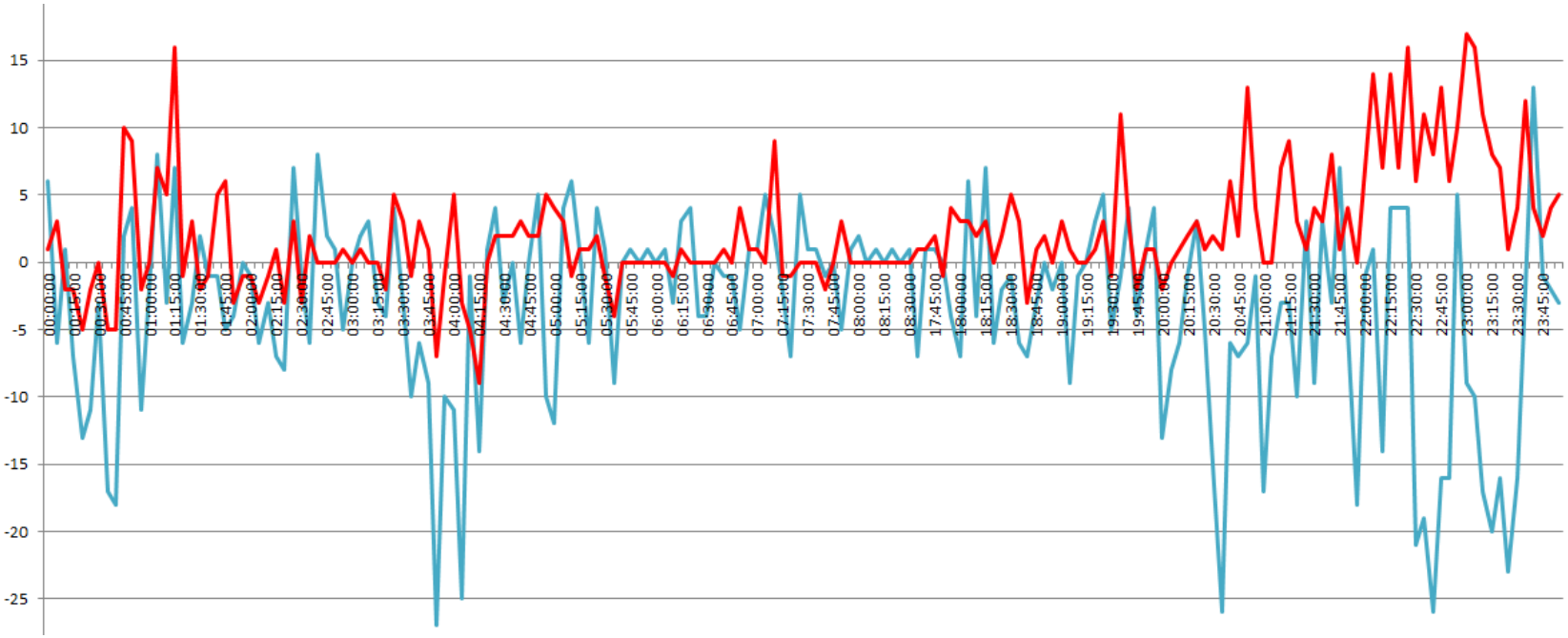

Figura 2.6 Andamento giornaliero utente C. Il grafico rosso rappresenta la polarità dei messaggi calcolata utilizzando solo le *emoticon* (come effettuato per la creazione del corpus **EMOLABEL**). Il grafico blu rappresenta la polarità calcolata con *RNNLM e word2vec*. Anche in questo caso il basso coefficente di correlazione tra i due grafici è dovuto alla forte mancanza di *emoticon*.

Capitolo 3

Conclusioni e sviluppi futuri

Giunto al termine di questo mio lavoro, vorrei ripercorrere i passaggi che ne hanno scandito lo sviluppo ed hanno tratteggiato gli aspetti e le problematiche relative alla *sentiment analysis*. Dopo un'introduzione teorica, in cui sono stati affrontati i temi della formalizzazione e della codifica della *text classification* ed è stata proposta una definizione del concetto di opinione, è stata effettuata una rassegna di tutte le metodologie attualmente utilizzate per eseguire un' analisi della *polarity*, per giungere, infine, alla progettazione di un sistema di *sentiment analysis* per chat, realizzato mediante l'utilizzo dei dati provenienti dalla piattaforma di video *streaming* Twitch.tv.

Per lo sviluppo applicativo si è fatto ricorso a due differenti modelli, il primo basato su *recurrent neural network* e il secondo impostato sull'utilizzo di una tecnica di *word embedding* tramite *word2vec*. Il sistema di classificazione proposto provvede ad effettuare una normalizzazione del testo sevendosi di un *tokenizzatore*, modificato in maniera opportuna per meglio adattarsi alle caratteristiche dei dati analizzati, e fornisce come output la polarità, espressa sotto forma di orientamento positivo o negativo per ogni frase della chat. Vista la totale mancanza di dati già etichettati provenienti dalla chat di Twitch, sono analizzate le venti emoticon più usate e, attraverso un processo di valutazione manuale, si è provveduto a definire due insiemi di emoticon, il primo contenente le emoticon positive ed il secondo contenente quelle ne-

gative. Per mezzo di questa categorizzazione il sistema è stato impostato su un sottoinsieme di dati raccolti, per essere poi testato su un altro sottoinsieme degli stessi dati, categorizzati attraverso l'uso delle emoticon. In questo modo si è potuto constatare come il sistema riesca effettivamente a fornire, con livelli di precisione alti (circa il 76%), una netta suddivisione tra frasi contenenti messaggi positivi e frasi connotate da messaggi negativi. Si è poi passati all'esecuzione dei test, effetuandoli su un insieme di frasi categorizzate manualmente. Con questo metodo si è potuto stabilire che il sistema riesce concretamente a fornire una valutazione sulla polarità delle frasi, sia quelle contenenti *emoticon* sia quelle che ne risultano prive, procurando risultati migliori di quelli ottenibili utilizzando un approccio classico attraverso un modello Bag-of-Words. La parte sperimentale si è conclusa con l'analisi dell'andamento giornaliero di alcune trasmissioni: in questa fase l'attenzione è stata focalizzata sull'aspetto di riconoscimento di eventi (positivi o negativi) avvenuti nell'arco della giornata ed in particolare sulla capacità del sistema nel riuscire ad individuare tali eventi anche in mancanza di frasi contenenti *emoticon*. Infine si è giunti a dimostrare che, mentre un'analisi che consideri soltanto *emoticon* può portare a risultati fuorvianti, il modello realizzato in questa tesi rimane allineato ai valori di polarità realmente espressi dagli utenti della chat.

3.0.1 Sviluppi futuri

Nell'analisi dei dati e dei modelli sono state prese in esame diverse caratteristiche che vale la pena approfondire.

Frasi sarcastiche

Sono state individuate frasi contenenti sarcasmo, come riportato nel paragrafo 2.6: dall'analisi effettuata, purtroppo, tali frasi non sembrano essere legate a determinate *emoticon*, e non è quindi possibile effettuare una categorizzazione automatica. Prendendo a modello le numerose ricerche pubblicate

basandosi su dati provenienti da Twitter Wang et al. (2015) potrebbe essere interessante procedere ad uno studio analogo, con dati provenienti dalla chat.

Frasi riferite a particolari eventi

Un altro aspetto interessante emerso da questo studio, dovuto principalmente all'utilizzo massiccio di *emoticon* sulla piattaforma Twitch, consiste nel fatto che determinati argomenti, come per esempio il nome di un giocatore o quello di un particolare evento, vengono spesso utilizzati in frasi contenenti sempre la stessa emoticon. L'*emoticon* smorc  , ad esempio, viene associata a messaggi riferiti a giocatori particolarmente aggressivi, mentre volcania  , che rappresenta il simbolo di una particolare convention annuale svolta negli Stati Uniti, viene utilizzata per riferirsi all'evento stesso. L'individuazione del significato di queste emoticon, quindi, rende possibile categorizzare in maniera automatica tutti i messaggi che si riferiscono ad un particolare argomento e permette di utilizzare questi dati per analizzare il parere e le opinioni espresse dagli utenti della piattaforma *Twitch*.

Frasi riferite alla qualità della trasmissione

Un'ulteriore analisi effettuata sui dati ottenuti, infine, ha evidenziato come sia possibile procedere ad un'ulteriore indagine dei messaggi raccolti, sviluppandola su due livelli. In particolare, potremmo definire "primo livello " quello che rappresenta la polarità del contenuto della trasmissione: si è notato, ad esempio, come i comportamenti degli utenti della chat siano strettamente collegati agli eventi che si verificano: infatti, qualora avvenga un evento positivo (una vittoria) aumenta il numero dei commenti positivi, mentre nel caso in cui si determini una situazione negativa (una sconfitta) cresce il numero dei commenti negativi. Possiamo notare, inoltre, come sia presente anche un secondo livello, che può essere definito meta-livello, in cui le frasi scritte nella chat non sono più riferite ad eventi riguardanti i contenuti della trasmissione, poiché oggetto della discussione diventa la trasmissione stessa. Al riguardo si possono indicare i seguenti esempi: "Video quality is

very good!" [positiva] oppure "The song you are playing is amazing!" [positiva], "the stream is lagging very badly!" [negativa], e cosi via. Analizzare la polarità di queste frasi rende possibile ottenere informazioni riguardanti l'aspetto tecnico della trasmissione e, di conseguenza, delineare un quadro preciso del servizio fornito. Procedendo nell'analisi dei dati raccolti per sviluppare questa tesi, si è ritenuto opportuno soffermarsi ad esaminare anche questo aspetto. Da una prima osservazione è emerso che la raccolta di questi messaggi comporta due problemi principali: il primo consiste nella difficoltosa categorizzazione di queste frasi, dovuta alla necessità di ricorrere ad una valutazione manuale, visto che le stesse non presentano quasi mai delle *features* facilmente individuabili in maniera automatica. Il secondo problema nasce dal fatto che il numero di frasi contenenti opinioni riguardanti il meta-livello sono in numero assai inferiore rispetto a quelle che esprimono una polarità sui contenuti e questo rende necessario estendere l'analisi ad un numero molto elevato di frasi.

Bibliografia

Akira, U. (2000). Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Pragmatics*.

Asher, N., Benamara, F., and Mathieu, Y. (2008). Distilling opinion in discourse: a preliminary study. *Proceedings of International Conference on Computational Linguistics, COLING'08*.

Baroni, M. and Lenci, A. (2010). *Distributional memori: A general framework for corpus-based semantics*. Computational Linguistics.

Bo, P. and Lee, L. (2008). Using very simple statistics for review search: An exploration. *Proceedings of International Conference on Computational Linguistics, poster paper COLING-2008*.

Cheng-Yu, L., Shian-Hua, L., Jen-Chang, L., Samuel, C.-L., and Jen-Shin, H. (2010). Automatic event-level textual emotion sensing using mutual action histogram between entities. *Journal of Expert Systems with Applications*.

Chenlo, J., Hogenboom, A., and Losada., D. (2013). Sentiment-based ranking of blog posts using rhetorical structure theory. *Presented at the 18th international conference on applications of Natural Language to Information Systems*.

Chien, C. C. and You-De, T. (2011). Quality evaluation of product reviews using an information quality framework. *Journal of Decision Support Systems*.

Christopher, M., Raghavan, P., and Schutze, H. (2008). Introduction to information retrieval. vol. 1. *Cambridge University*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning reserch*.

Ding, X. and Liu, B. (2010). Resolving object and attribute coreference in opinion mining. *Proceedings of International Conference on Computational Linguistics COLING-2010*.

Efstratios, K., Christos, B., Theologos, D., and Nick, B. (2013). Ontology-based sentiment analysis of twitter posts. *Journal of Expert Systems with Applications*.

Fahrni, A. and Klenner, M. (2008). Old wine or warm beer: target-specific sentiment analysis of adjectives. *In: Proceedings of the symposium on affective language in human and machine, AISB;*.

Francis, C. (2012). Sentiment analysis via dependency parsing. *Journal of Experimental Psychology: General*.

Georg, G. and Hauffa, J. (2011). Characterizing social relations via nlp-based sentiment analysis. *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM-2011)*.

Gibbs and Raymond, W. (2015). On the psycholinguistics of sarcasm. *Experimental Psychology: General*.

Gibbs, Raymond, W., and Colston, H. L. (2007). *Irony in language and thought: A cognitive science reader*. Lawrence Erlbaum.

González-Ibáñez, Muresan, S., and Wacholder., N. (2011). Identifying sarcasm in twitter: a closer look. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

Hanhoon, K., Joon, Y. S., and Dongil, H. (2012). Senti-lexicon and improved naive bayes algorithms for sentiment analysis of restaurant reviews. *Journal of Expert Systems with Applications.*

Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. *In: Proceedings of annual meeting of the Association for Computational Linguistics (ACL'97).*

Hatzivassiloglou, V. and Wiebe, J. (2000). Effects of adjective orientation and gradability on sentence subjectivity. *Proceedings of Interntional Conference on Computational Linguistics.*

Heerschop, B., Goossen, F., Hogenboom, A., Frasincar, F., Kaymak, U., and de Jong, F. (2011). Polarity analysis of texts using discourse structure. *Presented at the 20th ACM Conference on Information and Knowledge Management (CIKM11).*

Hu, Y. and Wenjie, L. (2011). Document sentiment classification by exploring description model of topical terms. *Journal of Computer Speech and Language.*

Hye-Jin, M. and Jong, P. (2012). Identifying helpful reviews based on customer's mentions about experiences. *Journal of Expert Systems with Applications.*

Isa, M. and Piek, V. (2012). A lexicon model for deep sentiment analysis and opinion mining applications. *Journal of Decision Support Systems.*

Jian, J. and Yanquan, Z. (2011). Sentiment polarity analysis based multi-dictionary. *In: Presented at the 2011 International Conference on Physics Science and Technology (ICPST'11).*

Jonathan, O.-H., Diego, R. J., Leandro, A., Manuel, L., Inaki, I., and Jose, L. (2012). Approaching sentiment analysis by using semi-supervised learning of multi-dimensional classifiers. *In Proceedings of Neurocomputing 2012.*

Kaufmann, J. (2012). Jmaxalign: A maximum entropy parallel sentence alignment tool. *Proceedings of International Conference on Computational Linguistics, COLING'12.*

Kim, S. and Hovy, E. (2004). Determining the sentiment of opinions. *Proceedings of interntional conference on Computational Linguistics COLING'04.*

Kreuz, A., Roger, J., and Glucksberg., S. (1989). How to be sarcastic: The echoic reminder theory of verbal irony. *Experimental Psychology: General.*

Kreuz, E., Roger, J., and Caucci, G. M. (2007). Lexical influences on the perception of sarcasm. *Proceedings of the Workshop on Computational Approaches to Figurative Language.*

Le, Q. V. and Mikolov, T. (2014). Distributed representations of sentences and documents. *31th International Conference on Machine Learning.*

Li, S. and Tsai, F. (2011). Noise control in document classification based on fuzzy formal concept analysis. *Presented at the IEEE International Conference on Fuzzy Systems (FUZZ).*

Liu, B., Yang, E., Huang, X., An, A., and Yu, X. (2007). Arsa: a sentimentaware model for predicting sales performance using blogs. *Proceedings of ACM SIGIR Conf. on Research and Development in Information Retrieval.*

Loeu, C. F., Jose, T., Fernando, E., Ortega, J., and Gaoeu, V. C. (2013). Long autonomy or long delay? the importance of domain in opinion mining. *Journal of Expert Systems with Applications.*

Melloncelli, D. (2012). *Sentiment Analysis in Twitter.* PhD thesis, Universita' di Bologna.

Mesnil, G., Mikolov, T., and Bengio, Y. (2015). Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *Proceeding of the International Conference of learning Representations.*

Mikolov, T. (2012). *Statistical language models based on neural networks.* PhD thesis, Faculty of Information Technology.

Mikolov, T., Corrado, G., Chen, K., and Dean, J. (2013a). Efficient estimation of word representations in vector space.

Mikolov, T., Corrado, G., Sutskever, I., Chen, K., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Proceeding of the International Conference of learning Representations.*

Mikolov, T., Deoras, A., Kombrink, S., Burget, L., and Cernocky, J. (2011a). Empirical evaluation and combination of advanced language modeling techniques. *In Proceedings SpeechFIT.*

Mikolov, T., Karafiat, M., Burget, L., cernocky, J., and Khudanpur, S. (2010). Recurrent neural networks based language model. *In Proceedings of Interspeech.*

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., and khudanpur, S. (2011b). Extensions of recurrent neural network language model. *Journal of IEEE Xplore.*

Minqing, H. and Liu, B. (2004). Mining and summarizing customer reviews. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. *Proceedings of the international workshop on artificial intelligence and statistics.*

Nan, H., Indranil, B., Sian, K. N., and Ling, L. (2012). Manipulation of online reviews: an analysis of ratings, readability, and sentiments. *Journal of Decision Support Systems.*

Pak, A. and Paroubek, P. (2013). Twitter corpus for sentiment analysis and opinion mining. *Journal of Decision Support Systems.*

Ramanathan, N., Liu, B., and Choudhary., A. (2009). Resolving object and attribute coreference in opinion mining. sentiment analysis of conditional sentences. *Conference on Empirical Methods in Natural Language Processing (EMNLP-2009).*

Read, J. and Carrol, J. (2013). Weakly supervised techniques for domain-independent sentiment classification. *Journal of Decision Support Systems.*

Riloff, Ellen, S. P. and Wiebe., J. (2006). Feature subsumption for opinion analysis. *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Roberts, K., Roach, M. A., Johnson, J., Guthrie, J., and Harabagiu, S. M. (2010). Empatweet: Annotating and detecting emotions on twitter. *Journal of Decision Support Systems.*

Rodrigo, M., Francisco, V. J., and Wilson, G. N. (2013). Document-level sentiment classification: an empirical comparison between svm and ann. *Journal of Expert Systems with Applications.*

Tae, Y. and Smith, N. A. (2010). What's worthy of comment? content and comment volume in political blogs. *Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM 2010).*

Turney, P. D. (2012). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *Journal of Decision Support Systems.*

Wang, Z., Wu, Z., Wang, R., and Renn, Y. (2015). Twitter sarcasm detection exploiting a context-based model. *Web Information System Engineering.*

Wille, R. (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. *Proceedings of ACM SIGIR Conf. on Research and Development in Information Retrieval.*

Youngjoong, K. and Jungyun, S. (2000). Automatic text categorization by unsupervised learning. *Proceedings of International Conference on Computational Linguistics, COLING-00.*

Youngjoong, K. and Jungyun, S. (2010). Automatic text categorization by unsupervised learning. *Journal of Decision Support Systems.*

Yu, H. and Hatzivassiloglou, V. (2003). Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2003).*

Yulan, H. and Deyu, Z. (2011). Self-training from labeled features for sentiment analysis. *Inf Process Manage.*

Zhai, E., Zhongwu, B., Liu, B., Xu, H., and Jia, P. (2010). Grouping product features using semi-supervised learning with soft-constraints. *Proceedings of International Conference on Computational Linguistics.*