Sommario

I servizi di microblogging come Twitter sono in costante diffusione e vengono utilizzati sempre più spesso per condividere opinioni e commenti. In questo lavoro di tesi abbiamo estratto un insieme di tweet relativi ad un catalogo di film, con l'obiettivo di classificarli in relazione al film oggetto del commento.

In questo processo, denominato *item detection*, abbiamo sperimentato la classificazione dei brevi e rumorosi tweet con diversi approcci, il principale basato su classificatori one-class SVM. La migliore con?gurazione ha portato ad ottenere, per un dataset composto da 40 item (con 800 tweet ciascun item), una precision pari a 92% a fronte di una recall di circa 65%.

Indice

Capitolo 1

Introduzione

La recente diffusione dei Social Network ha modificato radicalmente l'uso che gli utenti fanno di Internet. Tali strumenti permettono di scrivere e diffondere informazioni in modo semplice e immediato, al punto che sempre più persone condividono (con gli altri utenti del proprio Social Network) informazioni, più o meno personali, riguardanti il luogo in cui si trovano, che attività stanno svolgendo, persino cosa pensano. Ad esempio, secondo uno studio (condotto su utenti americani) effettuato da Yahoo e Nielsen [1], l'86% degli utenti usa uno smartphone mentre guarda la TV, di cui il 40% per navigare il sito di un Social Network, ad esempio esprimendo un giudizio sull'episodio della serie televisiva in corso.

In questo contesto, uno dei servizi di social networking più diffuso è Twitter: in questo sistema ognuno è identificato come utente con un proprio profilo personale e una bacheca sulla quale vengono pubblicati dei commenti (i *tweet*) riguardo qualsiasi argomento risulti interessante. Tramite un meccanismo denominato *following* è possibile ricevere e condividere informazioni. La caratteristica principale di Twitter è la proprietà dei tweet, la cui lunghezza massima è di 140 caratteri; per questo motivo Twitter viene definito un servizio di **microblogging**.

Le informazioni contenute in un sistema come Twitter non sono strutturate e sono per loro natura rumorose (ad esempio l'uso di slang o abbreviazioni), rendendo l'analisi del contenuto molto più complicata.

L'obiettivo di questa tesi è quello di costruire un sistema in grado di effettuare *item detection*, ovvero di analizzare il contenuto informativo dei singoli tweet e classificarli rispetto ad un insieme di item noto a priori. Dato un catalogo di item (ad es. film) vogliamo innanzitutto individuare

[1] http://advertising.yahoo.com/article/the-role-of-mobile-devices-in-shopping-process.html

quali di questi item sono stati discussi nei tweet di un utente. La fase di item detection rappresenta quindi il primo passo di un possibile scenario più ampio, il cui obiettivo è quello di catturare gli interessi di un utente, ovvero apprenderne il *profilo* (Figura 1.1).

Per costruire un profilo utente, è necessario, oltre a conoscere quali argomenti vengono trattati nei tweet, analizzarli nel loro complesso per ricavare un'opinione tramite tecniche di sentiment analysis. Questo tipo di sistema potrebbe essere integrato con un sistema di raccomandazione oppure con un sistema di targeted advertising, dove le informazoni sugli interessi dell'utente sono il requisito principale.

Questo lavoro di tesi si focalizza sul primo passo di questo processo, l'*item detection* nel contesto del *microblogging*. L'attività può essere sintetizzata nelle seguenti fasi: (i) estrazione dei tweet, (ii) analisi delle feature, (iii) definizione dell'algoritmo base, (iv) definizione dell'algoritmo SVM e infine (v) integrazione delle due metodologie.
Estrazione di un dataset di tweet relativi ad un elenco di item, utilizzando le API di Twitter. A causa delle particolari politiche di Twitter, non si trovano dataset disponibili ed è perciò stato necessario estrarre manualmente i tweet. Il dominio scelto è stato quello dei film, essendo particolarmente diffusi i tweet. Data la *pagina* Twitter di un film, abbiamo estratto i tweet relativi, assumendo che tutti parlassero di quello specifico film. In totale sono stati estratti 40 film con una media di 800 tweet ciascuno.
Analisi delle possibili feature (es. hashtag, unigrammi ecc.) da utilizzare per descrivere il contenuto dei singoli tweet.
Definizione di un primo sistema di classificazione basato sul semplice *keyword matching*, utilizzando solo l'informazione sul titolo del film. Questo approccio è stato denominato *algoritmo base*.
Definizione e analisi di un sistema basato su una batteria di classificatori one-class SVM. In questo approccio, denominato *algoritmo SVM*, viene appreso un classificatore one-class SVM per ciascun item presente a catalogo. La scelta di questa particolare tipologia di SVM è stata dettata dalla natura del dominio. Infatti in un classico sistema (es., Video On Demand) si ha a che fare con un catalogo di item (es., film) ampio (decine di migliaia) e dinamico (film quotidianamente aggiunti e rimossi). Mentre con un classico classificatore multiclasse dovremmo riapprendere l'intero modello ad ogni modifica dell'insieme di item, con un insieme di classificatori a singola classe è sufficiente apprnedere un nuovo classificatore per ciascun nuovo item.
Integrazione dell'algoritmo base e dell'algoritmo SVM. In questa fase è stata definita anche una politica di aggregazione dei risultati dei singoliclassificatori. Infatti, la presenza di un insieme di classificatori a songola classe può

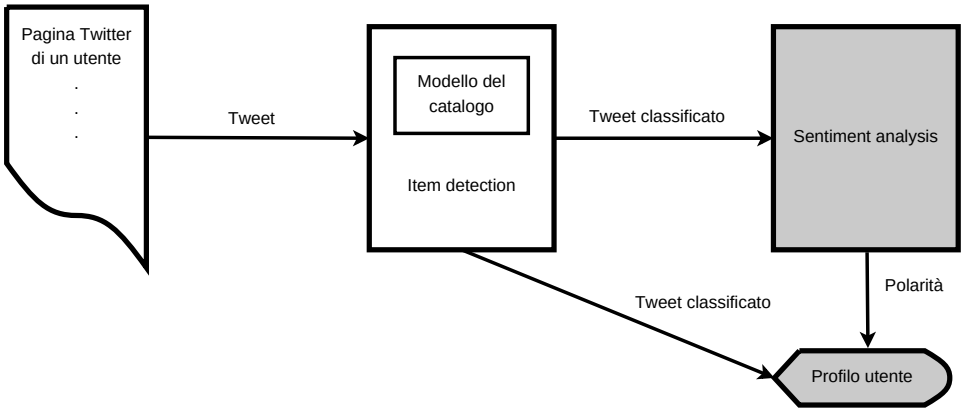


Figura 1.1: Schema generale del processo (le parti in grigio non sono state sviluppate in questa tesi)

portare ambiguità nell'assegnazione di un tweet ad una classe (ovvero ad un item/film), e in tal caso occorre stabilire una strategia di scelta.

Il sistema di classificazione di base, nella sua versione migliore, ha raggiunto delle prestazioni sorprendentemente buone, con una precision del 96% a discapito di una recall del 58%. Il sistema basato sui soli classificatori SVM, ha raggiunto le prestazioni migliiori utilizzando feature relative al titolo, agli hashtag e ai bigrammi, ma non è stato in grado di migliorare l'algoritmo base. Abbiamo infine sperimentato un approccio misto, in cui potessimo sfruttare le caratteristiche di entrambi i sistemi, ottenendo un aumento del 7% della recall a scapito di una diminuzione solo del 4% della precision, rispetto all'algoritmo base.

Struttura della tesi. Nel Capitolo 2 viene analizzato lo stato dell'arte relativo alle possibili fasi di un sistema completo, per avere una visione d'insieme del processo e capire il livello di ricerca in ciascun ambito. Nel Capitolo 3 viene formalizzato il problema: dalla costruzione del dataset, ai metodi di valutazione ai dettagli tecnici. Nei capitoli successivi vengono mostrati i diversi approcci con i relativi risultati: nel Capitolo 4 vedremo l'algoritmo di base, nel 5 analizzeremo quello basato sugli SVM e infine nel Capitolo 6 mostreremo 2 algoritmi avanzati. Dopo l'analisi completa, nel Capitolo 7 cerchiamo di trarre delle conclusioni globali e proponiamo diversi spunti sia per completare l'analisi approfondita di questi metodi, sia per estendere e eventualmente migliorare il nostro sistema.

Capitolo 2

Stato dell'arte

Il processo che trasforma il testo libero di un microblog in rating utili per un sistema di raccomandazione è molto articolato ma, nonostante questo lavoro ne sviluppi e approfondisca solo una parte, tutte le possibili fasi sono state esplorate inizialmente per avere una visione d'insieme e capire quale passo dell'intero processo non è ancora stato approfondito o addirittura trattato.

Nelle prossime sezioni vedremo quali altri lavori trattano i vari temi, o quelli simili, dell'intero processo seguendo il percorso logico dei dati e prestando una maggior attenzione alla parte sviluppata in questa tesi. La Sezione 2.1 spiega quali siano le tecniche di base per l'analisi del testo dal punto di vista sintattico e grammaticale, con diversi metodi che possono essere utilizzati nell'ambiente rumoroso dei microblog. Successivamente, nella Sezione 2.2 troviamo i primi passi nell'analisi del significato in senso puro, con i sistemi di base per la classificazione semantica. L'analisi più approfondita della classificazione del testo in categorie generiche è trattata nella Sezione 2.3 dove sottolineiamo la differenza tra i metodi per testi brevi e lunghi; infine un accenno ai pochi documenti esistenti nell'ambito dell'item detection, ovvero quello che vuole essere il tema principale di questa tesi, con le sue peculiarità e difficoltà. La Sezione 2.5 si occupa della sentiment analysis, un settore dell'analisi del testo più avanzato e concettualmente successivo alla fase di item detection proposta in questa tesi; anche in questo caso è utile sottolineare le differenze tra i lavori su testi di tipo diverso. Infine nella Sezione 2.6 ci occupiamo di trattare le basi teoriche dell'algoritmo di classificazione utilizzato per la nostra item detection.

2.1 Natural language processing (NLP)

Con il nome di natural language processing (NLP) si intendono tutte quelle tecniche di intelligenza artificiale che lavorano su testo libero e che cercano di analizzare il contenuto e estrarre informazioni da esso. Le varie categorie di tecniche di NLP comprendono anche analisi approfondite del significato del testo stesso e, vista l'enorme dimensione del campo applicativo, includono algoritmi molto generici e diversi tra loro. In questa sezione intendiamo però mostrare quale siano le tecniche di base esistenti per la fase di pre-processing del testo e vedremo quali possono essere utilizzate nel nostro particolare campo. Molti algoritmi eseguono analisi a livelli di granularità diversa per cercare di superare il limite dell'utilizzo di una parola solo come voce di un vocabolario, cercando di contestualizzarla per ottenere più significato da essa.

I primi passi in questa direzione sono volti allo sviluppo del **Part-of-speech (PoS)** tagging, ovvero del riconoscimento di ogni singola parola come entità grammaticale inserita nella struttura della frase. In (Charniak et al. [14]) vengono analizzati alcuni modelli esistenti (in particolare il modello Markoviani) per l'apprendimento dei tag con particolare attenzione alla struttura delle equazioni che vengono utilizzate. Per lo stesso scopo vengono poi utilizzati molti tipi di modelli come i modelli basati su regole (Brill [11]), i modelli Markoviani (Cutting et al. [20]) (Brants [10]) e quelli massima entropia (Ratnaparkhi [63]). Nel paper di Gimpel et al. [26] vi è il primo approccio PoS al mondo dei mircoblog con accuratezza vicina al 90%; tale approccio è basato su un conditional random field in cui vengono utilizzate diverse (e una normalizzazione basata sulla fonetica) per tenere in considerazione molte strutture particolari in Twitter: gli hashtags, le mentions, gli URL, e l'utilizzo non standard del maiuscolo. Questo tipo di analisi può essere molto utile in fasi avanzate come la sentiment analisys mentre pare decisamente meno applicabile per il riconoscimento di un topic o addirittura di un item.

Altre fasi di pre-processing del testo utilizzate in questo settore sono principalmente legate alla **correzione** di errori e alla **pulizia** del contenuto che abbiamo a disposizione. In [85] [48] [81] [13] vengono proposti diversi metodi per la correzione degli errori nel testo e la sua pulizia in fase preliminare per migliorare le prestazioni del sistema successivo. Ad esempio, il primo mostra i metodi base di pulizia del testo a livello di singolo termine, il secondo formalizza il concetto di matching debole per i pattern di estrazione degli errori, oppure il terzo utilizza un sistema complesso per la pulizia e la normalizzazione del testo anche rispetto ad abbreviazione ed errori di com-

posizione del messaggio. Un metodo più avanzato è presentato da Hirst and Budanitsky in [33], in cui si utilizza tutta l'informazione possibile ricavata dal contesto della frase per cercare termini scorrelati rispetto al discorso.

L'ultimo task possibile che analizzeremo è quello della **normalizzazione** del testo. In questo settore esistono alcuni approcci di base come quello di Sproat et al. [72] in cui la normalizzazione è effettuata a livello di singolo termine, prendendo in considerazione i classici termini non standard come abbreviazioni, numeri, valute, date e così via. L'uso di queste tecniche è anche applicato a testi brevi come in (Cook and Stevenson[18]), pensato per l'analisi di SMS, oppure in (Han and Baldwin [29]) e (Kaufmann [40]) dove approcci simili sono testati proprio su Twitter. Uno dei diversi strumenti a disposizione per l'analisi e la misura di matching tra le stringhe è la **distanza di Levenshtein** [42], una misura standard per capire il matching tra due stringhe basata sul conteggio delle modifiche di caratteri singoli necessarie per trasformare una delle due stringhe nell'altra. Quasi la totalità del lavoro svolto in questi termini è pensato per analizzare testi lunghi, strutturati e scritti per facilitarne la lettura e carpire il contenuto da parte di tutti; questo tipo di approccio è inutilizzabile nel mondo del testo libero di un social network, ancora peggio se stiamo cercando informazioni in un microblog come Twitter. Anche le tecniche di pulizia e normalizzazione sono piuttosto avanzate per l'obiettivo dell'item detection; infatti dopo aver verificato le prestazioni di alcuni di questi approcci, abbiamo deciso di utilizzare una semplice pulizia basata su espressioni regolari e **stemming** (Porter [59]); una eventuale normalizzazione potrebbe addirittura far perdere la possibilità di apprendere termini che fanno parte dello slang o addirittura abbreviazioni diffuse e significative.

2.1.1 Analisi delle conversazioni

Un campo di ricerca importante soprattutto nel caso Twitter è quello dell'utilizzo delle conversazioni, che in questo social network sono implicite oppure esplicitate tramite il concetto di retweet in modi molto differenti tra loro, come evidenziato da Boyd et al. [9]. L'analisi delle conversazioni può essere sia un modo per la ricerca di tweet interessanti o per altri obiettivi, sia come analisi per la suddivisione della timeline di un utente in discussioni differenti. A tal proposito Yang et al. [83] propongono di analizzare lo spazio dell'utente e dei retweet per valutare un punteggio in grado di riassumere l'interesse che può avere un tweet stesso in modo globale. In (Welch et al. [79]) viene invece analizzata la differenza tra l'utilizzo del retweet e della mention, concludendo che il retweet è un indicatore di interesse mol-

to più forte del semplice concetto di follow; d'altra parte (Honeycutt and Herring[35]) analizzano come il concetto di mention sia la chiave per capire quanto l'utilizzo di Twitter sia anche fortemente influenzato dal concetto di conversazione, esplicitato proprio dal link diretto @. Per quanto riguarda l'analisi delle timeline per la suddivisione in conversazioni, un esempio di approccio è dato da Ritter et al. [66] in cui vengono utilizzati modelli di conversazione basati sull'analisi di feature come gli unigrammi (che ottengono le migliori prestazioni) per segmentare la conversazione; in una fase successiva si analizzano i topic delle varie sezioni per ricorstruire le conversazoni sull'assunzione che segmenti differenti (come quello di una domanda e quello di una risposta) possano essere in forte relazione se appartengono allo stesso topic. In (Lossio Ventura et al. [45]) vengono sfruttate invece molteplici informazioni per la ricostruzione delle conversazioni ossia quelle semantiche, quelle sociali e quelle temporali. Le informazioni semantiche sono estratte partendo da un preprocessing dei dati per pulire il contenuto del tweet, successivamente diverse misure di importanza, likelihood e similarità vengono valutate per i diversi tweet. Inoltre vengono calcolate anche misure di link tra gli utenti (sociali) e quelle temporali; tutti questi valori sono pesati per ottenere una misura della forza della correlazione tra due tweet e quindi di ricavare la suddivisione in conversazioni.

2.2 Analisi semantica

L'analisi semantica è il primo passo dell'analisi testuale che ci permette di utilizzare il contenuto come conoscenza. Nonostante i metodi siano basilari e fondati sulla statistica, e che l'argomento sia strettamente collegato alla Topic detection (§ 2.3), l'analisi semantica viene utilizzata solitamente con un'accezione più teorica. Uno dei primi obiettivi dell'analisi semantica è stato quello della rappresentazione della conoscenza per migliorare uno dei settori più cruciali in questo periodo: **l'information retrieval**. In (Pohorec and Zorman[58]) vengono utilizzate diverse tecniche di NLP (§ 2.1) per estrarre la conoscenza da testo già semi-strutturato, per inserirlo successivamente in un Database di conoscenza formale che potrà essere utilizzato da un algoritmo.

Molti lavori nell'ambito dell'analisi del significato (e non molto del contenuto) sono proposti per diversi scopi: in (Hatzivassiloglou and McKeown[31]) l'attenzione è puntata sulla **polarità** degli aggettivi; vengono utilizzate informazioni morfologiche per estrarre non solo gli aggettivi singolarmente ma anche quelli collegati da una congiunzione, infine i termini vengono clusterizzati, ottenendo anche la possibilità di far fronte alle congiunzioni che

inseriscono una negazione nel significato del testo. In (Hatzivassiloglou and Wiebe [32]) e (Riloff and Wiebe [65]) il focus è sulla **soggettività** di alcuni termini; mentre il primo si limita a valutare diverse misure per ricavare la polarità e la potenza degli aggettivi da utilizzare come inticatori di soggettività, il secondo si preoccupa ti utilizzare un vocabolario di termini preannotati per estrarre frasi soggettive con altissima precisione e utilizzarle per addestrare un algoritmo di pattern extraction che andrà a migliorare il sistema stesso. Infine Wiebe et al. [80] propongono di analizzare la posizione relativa (o anche consecutiva) di alcune parole nel testo per apprendere quali sono quelle più frequenti, mentre Jindal and Liu [37] analizzano come sia possibile estrarre e valutare le **frasi comparative** di diversi tipi (e di classificarle in base alla categoria) con un sistema basato prima su regole base e poi su metodi standard di apprendimento automatico.

Un altro settore tipico dell'analisi semantica è quello della **rappresentazione dei documenti**, per studiarne la similarità e per facilitare l'analisi del contenuto stesso. Uno dei metodi più famosi per questo compito è la **Latent Semantic Analysis (LSA)** (Landauer et al. [41]), ovvero un metodo matematico/statistico di rappresentazione del contenuto informativo di un gruppo di documenti; viene utilizzata un matrice che mette in relazione i termini del vocabolario con quelli che abbiamo a disposizione, e rappresenta i valori di questa relazione con una feature significativa (come il valore TF-IDF). Mentre LSA utilizza il metodo SVD per migliorare le prestazioni del sistema, Hofmann [34] costruisce un modello generativo ed effettua quella che si chiama probabilistic mixture decomposition; il metodo viene chiamato probabilistic latent semantic analysis e ottiene prestazioni migliori di LSA. Nel settore specifico dell'information retrieval, viene utilizzato un metodo leggermente diverso e più specifico che si chiama **Latent Semantic Indexing (LSI)** (Dumais et al. [22])

Il primo metodo per portare l'analisi semantica al livello della topic detection è la **Latent Dirichlet Allocation (LDA)** (Blei et al. [8]) in cui si utilizza un modello Bayesiano gerarchico e l'algoritmo di Expectation Minimization (EM) per rappresentare i vari documenti come un complesso mix diversi argomenti in base ad una probabilità a priori che i vari termini appartengano a determinati argomenti, rappresentata da una distribuzione di coefficienti. In questo modo vengono mischiati metodi statistici e generativi per ottenere migliori prestazioni e per mettere le basi per la classificazione di documenti in categorie basate sugli argomenti stessi. Anche in (Nigam et al. [52]) il focus è sulla classificazione di documenti utilizzando diversi metodi basati sull'algoritmo di EM per cercare di utilizzare documenti annotati come base di partenza per l'apprendimento, ma cercando di utilizzare

un modello in grado di inferire conoscenza anche da nuovi e non annotati documenti.

I metodi di analisi semantica visti in questa sezione sono serviti principalemte per capire quali tecniche vengono utilizzate per l'analisi del contenuto del testo; molti modelli di rappresentazione dei documenti che abbiamo visto possono essere utilizzati in modo leggermente modificato per adattarsi al mondo dei microblog, anche se l'ambiente di utilizzo rende necessaria una particolare attenzione nell'applicazione di alcuni di questi metodi, soprattutto quelli basati sulla statistica.

2.3 Topic detection

La differenza principale che vogliamo sottolieare nella divisione tra analisi semantica e topic detection sta nello scopo che diamo alla nostra ricerca di significato all'interno del testo: mentre nell'analisi semantica cerchiamo il significato principalmente per rappresentare la conoscenza, la topic detection si preoccupa di analizzare il contenuto di un documento per classificarlo come appartenente ad un certo tipo di topic oppure per cercare all'interno di esso sezioni che caratterizzino uno stesso argomento trattato da chi ha scritto quel documento. I metodi utilizzati in questo settore sono alla base di un ampio spettro di analisi che coinvolgono molte aree dell'inelligenza artificiale, ed è per questo che fare una divisione netta tra le categorie di approcci risulta complicato; basta pensare che anche le tecniche utilizzate nel campo della sentiment analysis (§ 2.5) sono molto vicine, se non derivate, da quelle che stiamo per elencare. Sebbene sia più facile distinguere tra sentiment e semantic analysis rispetto che tra semantic analysis e topic detection, abbiamo cercato di distinguerle in questo capitolo con le motivazioni appena citate. La prima differenza tra i vari tipi di algoritmi esistenti è ancora quella che ha caratterizzato tutta la ricerca bibliografica, ossia la lunghezza del testo che vogliamo analizzare.

2.3.1 Topic detection per testi lunghi

Un testo di dimensione normale come può essere una recnsione, un articolo, contiene moltissime informazioni testuali e inoltre è pensato per essere chiaro, esaustivo e in molti casi anche logicamente strutturato. La ricerca di un argomento all'interno di questo tipo di testi è un problema ben noto e sono molti gli approcci che vengono utilizzati nella pratica.

In Andrzejewski et al. [5] viene ripreso il concetto di LDA già visto nel paragrafo precedente e viene adattato per poter incorporare della cono-

scenza pregressa sui topic che abbiamo a disposizione. Nel paper sono noti quali siano i topic in questione e anche quali siano i link di correlazione tra le parole; questi link (sia positivi che negativi) sono codificati matematicamente e sostituiti ai coefficienti di Dirichlet a priori, classici della LDA. In questo modo non solo le parole vengono incluse nel modello, ma anche i collegamenti tra le parole stesse. Anche in (Andrzejewski and Zhu [4]) viene usato un metodo LDA modificato: in questo caso però la modifica dei coefficienti di Dirichlet viene inclusa in modo da poter essere modificata in modo supervisionato. In questo modo sarà possibile modificare a piacimento il valore del coefficiente relativo allo stesso termine nello stesso documento ma che appartiene chiaramente ad un topic differente, rendendo possibile una disambiguazione supervisionata.

Per quanto riguarda la topic detection all'interno di uno stream testuale, uno degli approcci più interessanti va sotto il nome di **Topic Detection and Tracking (TDT)**, di Makkonen et al. [47]. Lo stream di informazioni viene analizzato alla ricerca di eventi, ossia accadimenti specifici, i quali vengono rappresentati come vettori contenenti le parole significative correlate ad alcune classi semantiche. Per la ricerca del topic invece, si prendono tutti gli eventi e si clusterizzano utilizzando metriche di similarità diverse per ogni classe semantica, costruendo un modello di topic incrementale basato su tali eventi capace di classificare un nuovo e mai visto evento. Un approccio più classico e generico è spiegato da Li and Yamanishi [43], dove il documento da analizzare non è necessariamente uno stream sul quale effettuare un tracking dell'argomento. In questo caso si assume che ogni topic sia composto da alcuni termini principali noti e che tramite un processo di clustering dei termini si possa costruire un finite mixture model basato sui cluster per ogni testo. Utilizzando il precedente modello vengono cercati i punti nel testo in cui si parla di un certo argomento, si separa il testo in argomenti diversi e si ricava il topic da ogni segmento.

Un altra tecnica interessante è esposta da Griffiths et al. [28] in cui il lavoro di topic detection è migliorato utilizzando anche le informazioni sintattiche analizzate nel testo. Usando le relazioni a breve e lunga distanza all'interno di frasi sintatticamente collegate, si può migliorare l'accuratezza di un sistema di topic detection. Verso l'utilizzo di un sistema completamente non supervisionato si inserisce il metodo di Wartena and Brussee [78] in cui vengono sfruttate delle conoscenze pegresse sui topic per verificare le prestazioni di un sistema di topic detection in modo automatico. Vengono utilizzati i testi degli articoli di Wikipedia come input e tramite un algoritmo di clustering si dividono i termini nei diversi topic (sconosciuti). I risultati mostrano una certa coerenza tra i topic estratti e quelli dichiarati

da Wikipedia e inoltre viene testata una nuova misura di similarità migliore di quella basata sul coseno.

2.3.2 Topic detection per testi brevi

Il lavoro svolto nella direzione di adattare i metodi classici dell topic detection al mondo dei microblog sono stati diversi; il problema fondamentale è che l'unità minima di informazione che abbiamo a disposizione è molto corta, rendendo inefficace il solo approccio statistico. Nonostante ciò, prendendo in considerazione topic piuttosto generici è possibile utilizzare la sequenza di messaggi di una conversazione su Twitter ed analizzarla alla ricerca di un argomento. Quando la ricerca riguarda molti tweet o argomenti generici è possibile svolgere un primo lavoro come quello di Rowlands et al. [67] in cui vengono utilizzati gli URL tweetati dagli utenti, e per quelli più famosi o d'interesse, si costruisce una rappresentazione di tali link in base al contenuto semantico dei tweet in cui sono presenti. Questo tipo di approccio è il più semplice perchè il termine discriminativo per la ricerca e l'analisi sono gli URL (univoci) dei quali si cerca una spiegazione. Un altro mtodo per classificare i tweet (o gruppi di essi) in alcune categorie è mostrato da Ramage et al. in [62], dove gli autori cercano di utilizzare una tecnica chiamata labeled LDA per dividere i tweet di ogni utente in tipi di tweet. Le categorie sono soltanto cinque e la ricerca dei tweet è ancora semplice, in quanto stiamo analizzanto un singolo utente e sappiamo dove trovare i suoi tweet. Questi modelli vengono poi usati per confrontare profili dell'utente e raccomandare altri utenti da seguire.

Andando nella direzione della topic detection and tracking (Sakaki et al. [69]) mostrano nel loro paper come sia possibile utilizzare i tweet come se fossero uno stream di dati e cercare al loro interno eventi particolari. Visto il diffuso utilizzo di Twitter e la sua immediatezza, loro lo utilizzano come sensore di eventi. Infatti tweet di questo genere hanno particolari caratteristiche temporali, geografiche, stilistiche e di volume; gli autori sfruttano tutto questo insieme all'analisi semantica fatta costruendo un modello basato su keyword specifiche e correlate sia note a priori che ricavate proprio dai tweets, per generare features da classificare tramite una Support Vector Machine (SVM). In (Doerhmann [21]) viene utlizzato un altro metodo per ricercare tramite classificazione le **Named entity** ossia quelle parti del testo che sono riconduibili a categorie come persone, luoghi, organizzazioni. Il problema è ancora riconducibile a quello della classificazione in un numero molto basso di classi, infatti viene usato un metodo misto tra due approc-

ci esistenti e già noti: il Conditional Random Field (CRF) e il K-Nearest Neighbors (KNN).

Due approcci differenti sono interessanti nel campo della topic detection in uno stream di tweets per cercare argomenti interessanti ed emergenti nelle varie conversazioni. Il primo è di Cataldi et al. [12]: in questo paper si cerca di caratterizzare il **topic emergente** analizzando dapprima il suo contenuto semantico attraverso l'analisi dei termini contenuti nei tweets e poi anche l'influenza e le relazioni sociali che gli utenti hanno tra loro, cercando di ricavarne un andamento temporale e sociale. Analizzando poi il comportamento dei tweets nel tempo sono in grado di caratterizzare e trovare all'interno del flusso di tweets, gli argomenti emergenti. Un approccio diverso ma con lo stesso obiettivo è quello mostrato da Kasiviswanathan et al. [39] in cui si costruisce e analizza una dictionary matrix alla ricerca di documenti ïnuovi¿ basandosi sulla sparsità della matrice stsessa. Il passo successivo è quella di utilizzare tecniche di dictionary learning per clusterizzare i documenti considerati nuovi e rappresentare tali cluster come argomenti emergenti.

Il documento più interessante in questo settore, che si mescola con i sistemi di raccomandazione e che si avvicina all'approccio che viene usato in questa tesi è quello di Chen et al. [15] in cui l'obiettivo è quello di raccomandare URL presenti nei tweets. Il processo è diviso in tre parti: in una prima fase vengono scelti gli URL più interessanti, sia in base al fatto che siano popolari in tutto il mondo, sia in base alla loro frequenza in un particolare gruppo di persone correlate dalla relazione follower-followee. Una volta costruito l'insieme degli URL che vogliamo prendere in considerazione, vengono costruiti i modelli di tutti gli URL basandosi sull'analisi dei termini che compaiono nei tweets di ciascun utente che commenta o tweeta un URL. Allo stesso modo vengono costruiti i modelli per ciascun utente per capirne i topic d'interesse. Tutti questi modelli sono costruiti attraverso feature vector di coppie termine/entità caratterizzate dal valore TF-IDF e preprocessate da una fase di stemming. In una fase finale vengono utilizzati alcune combinazioni di questi passi per verificare la bontà di diversi sistemi di ranking per la raccomandazone finale. Come si può vedere da questo esempio, man mano che ci avviciniamo ad una topic detection in cui i topic diventano molti e specifici, l'utilizzo di Twitter diventa molto rumoroso e pericoloso, tanto da rendere necessario un passo indietro nell'analisi del testo e quindi una semplificazione delle procedure di analisi semantica. Anche nella Sezione sulla Sentiment analysis (§ 2.5) si presenterà questo problema, a sottolineare come andando a scontrarsi con problemi sempre più specifici nel campo della semantica, l'utilizzo di Twitter deve essere progettato per far fronte ai suoi limiti e all'utilizzo che gli utenti ne fanno.

2.4 Item detection e user modeling

Le ricerche nel campo della Item detection e dell'user profiling sono accomunate dalla necessità di costruire dei modelli di entità come possono essere items e utenti utilizzando fonti di informazioni non strutturate e fortemente rumorose come quelle di un microblog. La differenza principale sta nel fatto che per la costruzione del profilo utente abbiamo a disposizione una sorgente di informazione inequivocabile, in quanto gli ambienti dei microblog, come tutti i social networks, sono fortemente cartterizzati dalla sfera personale e dalla costruzione propria di un profilo basata sulle entità informative del social network. Nella modelizzazione di un item invece, siamo di fronte ad una nuova sfida: vogliamo caratterizzare con un modello un item, ovvero un'entità che spesso non ha sorgenti üfficialïdi informazione, in cui quindi il fattore rumorosità dell'ambiente esplode in maniera esponenziale, costringendoci ad essere il più conservativi possibile.

In Abel et al. [2] gli autori cercano un link semantico tra i tweet degli utenti e le news a cui sono collegate, cercando di utilizzare topic, entità, hashtags, URL e altre tecniche di base per fare inferenza statistica sul testo. Il risultato di quest'analisi è poi risultato molto utile nella costruzione di profili utente in grado di essere più specifici e descrivere meglio gli interessi dell'utente. Il settore delle news è stato molto esplorto in quanto gran parte dei tweet in circolazione riguardano notizie e sono strettamente legati ad un lasso temporale ben definito; inoltre è una caratteristica di Twitter quella di essere utilizzato specialmente per diffondere informazioni. Nello stesso settore troviamo il lavoro di Abel et al. [1] in cui vengono utilizzati i **profili utente** semanticamente arricchiti per collegarli ad una serie di argomenti predefiniti e raccomandare notizie relative agli argomenti d'interesse. In questo tipo di approcci viene fatta una forte analisi temporale per caratterizzare il comportamento delle notizia che vengono commentate in rete; infatti anche nel paper di Phelan et al. [57] troviamo un metodo interessante per raccomandare news. In questo caso il termine di confronto tra tweet e argomento della news viene ricavato dei feed RSS relativi ad un qualche argomento specifico oppure appositamente taggati. In seguito viene costruita una matrice delle co-occerenze in cui vengono utilizzati i valori TF-IDF dei termini comuni ai tweet che stiamo analizzando e alle notizie ricavate dai feed RSS per creare una classifica delle news e poterle raccomandare. Anche se l'ambiente in cui sono sviluppati questi lavori si avvicina sempre di più a quello descritto in questa tesi, siamo ancora lontani dall'avere un insieme di item sconosciuti che vogliamo modellizzare facendo ipotesi forti sulla sicurezza delle fonti a disposizione.

Con lo scopo di riorganizzare la timeline di un utente di Twitter in base ai propri interessi, Lu et al. [17] propongono di modellizzare i topic utilizzando le pagine di Wikipedia e costruendo i vettori di feature con i valori TF-IDF e allo stesso modo, costruire i vettori di feature dei tweet. Utilizzando l'algoritmo di Explicit semantic analysis (ESA) è possibile calcolare una misura di matching tra tweets e argomenti e, utilizzando i tweet di un utente, costruirne un profilo. Utilizzando poi le relazioni tra utenti (ricavate con ESA sui profili utente) e le informazioni sui link tra argomenti disponibili analizzando Wikipedia, è possibile condurre un random walk sul grafo dei link tra topic e sull'affinità tra utenti per scoprire i tweet candidati ad essere messi in primo piano. Un ranking basato sulla misura di matching tra tweet e topic viene utilizzato poi per ordinare i tweet per rilevanza.

Il lavoro che come obiettivo si avvicina di più a quello di questa tesi è quello di Wakamiya et al. [77] in cui gli autori cercano di ricavare informazioni da Twitter per migliorare le prestazioni di un **sistema di raccomandazione di programmi TV**. Come nel nostro ambiente, il problema fondamentale è quello di cercare all'interno dei tweets solo quelli rilevanti, ossia che parlano di programmi televisivi; come primo approccio vengono utilizzati gli hashtags e le informazioni geografiche per fare una prima scrematura. Come si vede è una necessità importante nel mondo dei microblog quella di fare un'attenta pulizia delle informazioni che compaiono sul social network. Il secondo step è quello di ricercare quegli utenti che stanno realmente parlando di uno specifico programma tv; per essere conservativi vengono usati metodi per ricercare il titolo o una parte di esso all'interno di ogni tweet usando sia gli hashtags che i dati provenienti dall'EPG. Infine per ricavare il programma TV di cui parla un tweet, vengono usate misure di similarità in tre dimensioni: testuale, spaziale e temporale. Infine, per creare un ordinamento di programmi TV per ogni utente e usare queste informazioni per migliorare il sistema di raccomandazone, si usa una misura di similarità mista ricavata dalle tre dimensioni.

2.5 Sentiment analysis

La sentiment analysis è quel processo per capire il senso di una frase o un documento dal punto di vista dell'opinione che l'autore vuole trasmettere con il documento stesso. Anche questo tipo di analisi è molto vasta e generica, infatti l'analisi del sentimento attraverso il testo può serivre a diversi scopi: possiamo essere interessati alla soggettività, alla polarità di una frase (positiva o negativa) piuttosto che proprio al tipo di emozione che traspare dal testo. Tutti questi obiettivi ricadono nella sentiment analysis in quanto

le modalità con cui raggiungiamo il nostro scopo sono simili; si cerca sempre dare più importanza a quelle parti di frase con cui le persone esprimono le sensazioni e le opinioni in un testo più o meno complesso. Nonostante gli approcci possano essere anche statistici, la peculiarità della sentiment analysis rende il compito difficile perchè ci spingiamo nel mondo dell'interpretazione del testo. Il vantaggio è però che le categorie con cui abbiamo a che fare sono spesso due o tre o una manciata di emozioni differenti (Wu et al. [82]), ed è per questo che può risultate spesso più semplice condurre una sentiment analysis su un testo breve piuttosto che una item detection (§ 2.4). Come è ben esposto in (Gong [75]) ci sono molti problemi da tenere in considerazione quando si effettua una sentiment analysis: dall'estrazione di frasi con opinioni, alla ricerca della soggettività, fino all'attribuzione di una polarita. Inoltre è indispensabile capire su quali parti del testo puntare; gli aggettivi sono i termini più utilizzati per esprimere emozioni ma anche avverbi e verbi possono dare il loro contributo. Da tenere bene in considerazione anche la relazione tra termini vicini e lontani all'interno delle frasi, spesso una semplice negazione modifica il significato dell'opinione. Analizzare tutto il contesto in cui le frasi sono poste è una sfida molto complessa, infatti a fianco ai metodi di classificazione sono anche molto diffusii metodi basati su un lessico predefinito, che supportano ma limitano l'apprendimento: è indispensabile trovare l'equilibrio giusto.

2.5.1 Sentiment analysis per testi lunghi

Il documento di riferimento per la sentiment analysis è sicuramente quello di Pang and Lee [55] in cui vengono analizzati moltissimi problemi di questa sfida: dalle feature da utilizzare, alle informazioni di contesto che possono essere usate, alle possibili soluzioni fino ai problem ancora da risolvere. Molte delle proposte e dei problemi affrontati in questo documento sono state molto utili nella scelta dell'approccio da seguire in questa tesi; nonostante la parte della sentiment analysis non sia stata sviluppata, la criticità e la necessità di essere conservativi al massimo mi ha dato notevoli spunti per trovare una possibile strada. Per questo motivo abbiamo proseguito con la ricerca sulla sentiment analysie, spostandomi anche sulle peculiarità dell'ambiente dei microblog.

Un approccio fortemente basato sulla **linguistica** e sulla **sintassi** è quello di Nasukawa and Yi [51] in cui le informazioni sintattiche delle frasi gli permettono di classificare le singole frasi in base alla polarità rispetto a classificare un intero documento. Viene usato un algoritmo di PoS e poi diverse regole basate sulla sintassi e su delle keyword note per estrarre la polarità di

tali frasi. Altri metodi di **machine learning** vengono proposti da Pang et al. [56] e Jin et al. [36]; il primo presenta tre modelli: un classificatore baysiano, un approssio a massima entropia e un SVM (§ 2.6.1), arrivando alla conclusione che, nonostante questi metodi non funzionino bene nella sentiment analysis almeno quanto nella topic detection (§ 2.3), il modello SVM è mediamente il migliore. Nel secondo documento viene invece proposto un approccio basato su un modello Markoviano. A partire da questi, sono molti gli approcci diversi per la sentiment analysis: da (Hasan and Adjeroh [30]) in cui viene proposto un approccio basato sull'analisi delle relazioni di prossimità tra i termini, a (Li et al. [44]) dove gli autori cercano di ottenere migliori prestazioni analizzando i documenti a livelli di granularità diversa e poi applicando un algoritmo diverso per ogni livello in grado di ottimizzare il risultato. Nel paper di Turney [76] l'autore utilizza le informazioni PoS e una misura di similarità chiamata PMI-IR per classificare i documenti come raccomandati o no, mentre Takamura et al. [73] cerca di migliorare le prestazioni della sentimenta analysis utilizzando un sistema di clustering a variabili latenti in grado di cogliere e contestualizzare la particolarità dello stesso termine utilizzata in situazioni diverse. Altri algoritmi più avanzati per la sentiment analysis includono: metodi basati su alberi e sequenze di termini (Matsumoto et al. [49]), metodi basati su SVM con feature arricchite (Mullen and Collier [50]), approcci ibridi [16] [60] e metodi adattivi semi supervisionati [71].

Una categoria importante di algoritmi di sentiment analysis è quella che riguarda la classificazione dei documenti in base alla **soggettività**. Infatti non è sempre detto che in un documento vi sia espressa un'opinione, moltissimi documenti potrebbero essere notizie, fatti oppure chatting. Il tipo di classificazione e le tecniche sono molto simili però in questi casi affrontiamo un problema leggermente diverso e che potrebbe essere di supporto ad un sistema più articolato. Un modo di utilizzare il precedente metodo di Esuli and Sebastiani [24] basato sull'espansione di un dizionario di termini positivi o negativi tramite l'uso dei sinonimi o delle definizioni dei termini in questione, è degli stessi Esuli and Sebastiani [70] in cui lo stesso tipo di espansione viene utilizzata per caratterizzare anche le categorie öggettivoë s̈oggettivo; infine analizzando la soggettività, la polarità e la forza di un'opinione arrivano alla conclusione che la ricerca della soggettività di un documento è un compito ancora più complesso. Yu and Hatzivassiloglou [84] propongono un'analisi multilivello utilizzando un classificatore Bayesiano utilizzando una misura di similarità basata su feature di tipo statistico. Questo passo dell'algoritmo arriva ad un'accuratezza tale che il lavoro è fatto proseguire nell'analisi della polarità dei documenti o delle frasi, con ottimi

risultati. In (Pang and Lee [54]) viene mostrato quale sia l'effetto di una precedente classificazione in base alla soggettività nei confronti di una sentiment analysis. Il classificatore di soggettività è un Bayesiano, mentre quelli di polarità possono essere sia Bayesiani che SVM, l'apprendimento è basato su un algoritmo di tagli su grafi. Il risultato è che il miglioramente più netto è quello di utilizzare un classificatore di soggettività prima di passare ad un classificatore di polarità Bayesiano.

Molti altri paper sono focalizzati sulla ricerca di un metodo automatico per **l'espansione di un vocabolario** in grado i essere utilizzato per una futura sentiment analysis. In (Kanayama and Nasukawa [38]) e (Qiu et al. [61]) l'obiettivo è quello di sfruttare le strutture sintattiche di un documento per cercare all'interno di un topic specifico quali siano i termini correlati e ordinarli in base ad un valore di polarità. Infine, nel primo documento con una soglia adattiva e nel secondo con un CRF sul quale viene effettuato un pruning intelligente, i migliori vengono aggiunti al dizionario di termini. Un altro metodo interessante è quello di Gamon and Aue [25] in cui, per l'espansione di un lessico legato ad un particolare contesto, non vengono utilizzate solo le relazioni forti di dipendenza tra i termini, ma vengono enfatizzate e utilizzate le debolissime relazioni che possono essere interpretate come negazioni di una relazione stessa e generare quindi un lessico completo anche di termini che sicuramente non appartengono al contesto.

2.5.2 Sentiment analysis per testi brevi

La sentiment analysis applicata a testi brevi ha sempre il problema della dimensione del testo a disposizione ma, come supposto e verificato da Bermingham and Smeaton [7] la concentrazione di informazioni utili in un testo così corto, rende più semplice la sentiment analysis. Gli utenti di Twitter sono abituati ad utilizzare il loro social network per scambiare messaggi che sono molto spesso opinioni o commenti; questo ambiente ci permette di avere una ricca collezione di documenti sui quali lavorare. Inoltre il vincolo dei 140 caratteri di Twitter costringe gli utenti ad inserire nel testo solo le informazioni indispensabili; questo comportamento rende Twitter un ambiente paradossalmente più semplice da analizzare per quanto concerne la sentiment analysis. Mentre nella topic detection (§ 2.3.2) o nella item detection (§ 2.4) questo comportamente è un problema perchè impoverisce il contesto di un tweet rendendo vano parte del lavoro, al contrario per la sentiment è un vantaggio perchè la maggioranza dei tweet conterrà in modo conciso e esplicito l'opinione in modo quasi inequivocabile. L'analisi di questo settore della sentiment analysis è già sviluppato, ma la conoscenza dello stato

dell'arte in questo campo e la sua efficacia nel caso particolare di Twitter, ci ha permesso di provare una nuova strada per l'item detection in modo da poter sfruttare proprio queste potenzialità per una futura integrazione con un sistema di raccomandazione. Un paper che si preoccupa di migliorare ancora di più l'ambiente di lavoro della sentiment analysis è quello di Saif et al. [68] in cui gli autori costruiscono sia feature semantiche (con un algoritmo di NER) sia feature che correlano opinioni e topic (utilizzando il Joint sentiment topic model) al fine di correlare i termini poco frequenti tra loro e migliorare le prestazioni del classificatore a valle (in questo caso un Bayesiano Naive). Sempre con lo scopo di ridurre la difficoltà del lavoro di un sistema di sentiment analysis Read [64] propone di utilizzare le frequentissime emoticons per costruire automaticamente un metodo per verificare in modo indipendente dal contesto quali termini possono essere correlati a differenti opinioni in contesti diversi.

I paper più significativi nell'ambito della sentiment analysis nel caso particolare di Twitter sono quasi tutti volti alla ricerca delle **features migliori** da utilizzare con un certo tipo di classificatore. Nonostante ci possa essere una differenza tra i vari tipi di classificatori a dosposizione, il focus è spesso sull'arricchimento delle features da utilizzare, in quando il mondo particolare di twitter offre moltissimi dati e metadati all'interno dei 140 caratteri disponibili. In (Agarwal et al. [3]) il testo dei tweet viene utilizzato e parsato alla ricerca di emoticons e acronimi per rendere la creazione delle features più efficace, in seguito vengono testati su un classificatore SVM diversi tipi di features: dagli unigrammi, alle features batase su un tree kernel in grado di condensare sintassi e PoS all'utilizzo di feature basate su un dizionario. Nel paper di Çelikyilmaz et al. [23] invece ogni tweet viene normalizzato rispetto ai valori numerici, agli URL, gli utenti, le named entity e vengono poi comparate le prestazioni di un approccio LDA. Inoltre mostrano come sia possibile aumentare le prestazioni utlizzando un clustering dei termini basato sulla pronuncia. Sempre utilizzando tutti i dati possibili all'interno di Twitter e sfruttando anche il profilo utente, Luo et al. [46] propongono di utilizzare un metodo di analisi del comportamento e della struttura dei tweet e dei retweet per classificare come pseudo-oggettivi e pseudo-soggettivi alcuni termini e includere queste informazioni nelle feature per creare un metodo automatico di riconoscimento di tweet con opinioni. Pak and Paroubek [53] utilizzano un metodo basato sulle emoticons per estrarre automaticamente un dataset da Twitter classificato in tweet positivi e negativi. Il dataset viene poi pulito da URL e caratteri speciali e poi vengono usati un tokenizer e un PoS tagger per costruire delle feature basate interamente sul testo come unigrammmi, bigrammi e trigrammi. Infine Barbosa and Feng [6] cercano

di costruire feature più astratte di quelle basate sui termini contenuti nel tweet per far fronte all'ambiente molto rumoroso di Twitter.

2.5.3 Trend analysis

Un settore molto sviluppato della sentiment analysis è quello della trend analysis, in cui l'obiettivo di ricercare le opinioni nel testo dei tweet è perseguito solo per un particolare argomento. Da questo punto di vista, ci avviciniamo a quello che è stato sviluppato in questa tesi: cerchiamo dei tweet specifici riguarda ad un topic per poi esrguirne una sentiment analysis. Gli approcci esistenti sono molteplici e sono spesso basati sulla ricerca di tweet per keyword, un metodo che però non è molto efficace e soprattutto non si adatta al comportamento che gli utenti di Twitter hanno mentre scrivono un commento. La trend analysis si basa sul fatto che non ci interessa chi ha scritto i tweet che riceviamo dalla nostra query perchè lo scopo finale è ottenere una statistica sulle opinioni correnti riguardo per esempio una certa organizzazione. Inoltre per moltissime organizzazioni o brand, la keyword è sufficiente per avere una certa confidenza nei tweet ottenuti.

Basandosi su queste considerazioni, sono stati sviluppati moltissimi tool online che propongono questo tipo di statistiche; ovviamente gli eventuali errori nel matching di tweet interessanti è visibile ma non evidente vista la natura statistica dell'approccio. A tal proposito è interessante il lavoro di Go et al. [27] in cui vengono analizzate molte combinazioni differenti di approcci al problema. Il primo metodo utilizzato è un classificatore Bayesiano addestrato con diversi tipi di feature basate sulla frequenza dei termini e sull'appartenenza di alcuni termini alle classi corrispondenti. Con lo stesso metodo viene provato un approccio a massima entropia, ottenendo le stesse prestazioni ma con un tempo di elaborazione e training considerevolmente maggiore. Un classificatore SVM viene poi utilizzato con le classiche feature statistiche come unigrammi, bigrammi, PoS e l'inclusione della negazione come feature. In questo caso le prestazioni non sono al livello del classificatore Bayesiano ma diverse considerazioni fanno pensare che la scelta dei parametri e del tipo di kernel potrebbe migliorarne molto le prestazioni.

2.6 One-class classification

L'obiettivo di un classificatore è quello di apprendere la distribuzione di punti appartenenti a classi diverse in uno spazio multidimensionale per ricavarne un modello o un metodo per discriminare in modo efficace qualsiasi punto sconosciuto ed associarlo alla sua classe reale. Tutti i metodi hanno in comu-

ne il fatto che l'apprendimento di tali regole discriminative deve rispettare un difficle equilibrio: da una parte vogliamo evitare che i modelli generati siano troppo vicini a quello che è il training set (overfitting), dall'altra vogliamo evitare di esagerare con la generalizzazione (underfitting). Solitamente il problema è ricondotto alla classificazione in N classi di cui conosciamo un set di punti uniformemente distribuito sulle varie classi. Tali metodi sono molto discriminativi ma presentano almeno due grosse limitazioni:

- è matematicamente complesso lavorare in ambienti con moltissime classi e l'accuratezza di tali sistemi ne risente moltissimo
- non è possibile attribuire ad un nuovo punto la non appartenenza a nessuna delle N classi

Per far fronte a questi due problemi che sono evidenti e reali nel nostro campo applicativo, abbiamo deciso di utilizzare un metodo di classificazione diverso, ovvero la One-class classification. Con tale metodo è possibile risolvere i problemi spracitati in questo modo:

- ogni classificatore one-class si comporta in modo uno vs. tutti, ossia un punto può appartenere o no alla classe in questione
- utilizzando in parallelo più classificatori one-class è possibile simulare il comportamento di un classificatore a N classi e ottenere il risultato di non appartenenza a nessuno di questi

Infine, un ulteriore vantaggio di questo approccio è che i classificatori one-class funzionano in presenza di punti del training set appartenenti solo alla classe target, non dobbiamo quindi fornire controesempi (che sarebbero moltissimi e difficili da ottenere un modo uniforme). Nelle prossime sezioni analizzeremo questo problema con l'utilizzo di Support Vector Machines (§ 2.6.1) e Support Vector Description Data (§ 2.6.2) che, fusi insieme ci permettono di utilizzare quello che chiameremo classificatore SVM One-class (§ 2.6.3).

2.6.1 Support Vector Machine

Una Support Vector Machine (SVM) è un modello per la classificazione e regressione ideato per separare linearmente due classi. In seguito alla sua diffusione sono stati proposti metodi per estendere il suo funzionamento a N classi e per rendere l'aproccio più generale, senza dover necessariamente utilizzare una separazione lineare. Per i nostri scopi è sufficiente l'utilizzo

di superfici non lineari di separazione, perchè la limitazione alle due classi non è un vincolo (nel caso One-class visto nella Sezione 2.6).

L'algoritmo parte dal già noto problema della ricerca dell'iperpiano ottimale di separazione tra due classi. I valori $\mathbf{x}_1$ del training set vengono rappresentati come vettori etichettati

$$(y_1, \mathbf{x}_1), \cdots, (y_l, \mathbf{x}_l), \quad y_i \in \{-1, +1\} \tag{2.1}$$

Tali punti sono linearmente separabili se esistono un vettore $\mathbf{w}$ e uno scalare b tali che per ogni punto in (2.1) venga rispettato il vincolo

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \cdots, l \tag{2.2}$$

Per trovare l'iperpiano ottimale

$$\mathbf{w}_0 \cdot \mathbf{x} + b_0 = 0 \tag{2.3}$$

dobbiamo cercare i valori di $\mathbf{w}_0$ e b_0 che massimizzano il margine di separazione dell'iperpiano per tutti i punti (2.1) sotto tutti i vincoli (2.2). Tale problema è noto come programmazione quadratica e la sua soluzione è nota ed efficiente.

I vettori $\mathbf{x}_i$ per cui $y_i (\mathbf{w} \cdot \mathbf{x}_i + b) = 1$ sono chiamati vettori di supporto e, inserendo i vincoli di tali vettori nel problema di programmazione quadratica con il metodo dei moltiplicatori di Lagrange, è possibile rappresentare l'iperpiano $(\mathbf{w}_0, b_0)$ in termini di tali vettori

$$\mathbf{w}_0 = \sum_{i=1}^{l} y_i \alpha_i^0 \mathbf{x}_i \tag{2.4}$$

dove gli α_i^0 sono i moltiplicatori di Lagrange dei vettori di supporto.

In (Cortes and Vapnik [19]) gli autori propongono di migliorare questo algoritmo utilizzando un metodo per tenere in considerazione la possibilità di trovare un iperpiano per i problemi contenenti **errori nel training set**, rendendo il sistema molto più robusto. Per formalizzare questa possibilità vengono introdotte alcune variabili $\xi_i \geq 0, i = 1, \cdots, l$ e viene considerata la funzione di errore

$$\Phi(\xi) = \sum_{i=1}^{l} \xi_i^{\sigma} \tag{2.5}$$

che per valori piccoli di σ, e minimizzata rispetto ai vincoli

$$y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \cdots, l \tag{2.6}$$

$$\xi_i \geq 0, i = 1, \cdots, l \tag{2.7}$$

ci permette di ricavare quali punti ci impediscono di separare linearmente lo spazio. Per trovare l'iperpiano che include anche tali punti con l'errore più piccolo dovremo minimizzare la funzione

$$\frac{1}{2}\mathbf{w}^2 + CF\left(\sum_{i=1}^{l} \xi_i^{\sigma}\right) \tag{2.8}$$

rispetto ai vincoli (2.6) e (2.7), dove C è una costante e $F(u)$ è una funzione monotona convessa. Il piano ricavato da questo algoritmo con $\sigma = 1$ è chiamato soft margin hyperplane ed è quello che separa le due classi con il minimo errore.

Il problema della linearità viene infine superato utilizzando una funzione kernel del tipo

$$K(\mathbf{x}, \mathbf{x}_i) \tag{2.9}$$

al posto dei singoli valori $\mathbf{x}$, in cui $\mathbf{x}_i$ rappresenta uno dei vettori di supporto. Tale funzione permette di proiettare i punti in uno spazio a dimensionalità più alta in cui sono linearmente separabili. Questo metodo viene chiamato **kernel trick**.

2.6.2 Support vector data description

Il primo esempio di classificazione one-class è mostrato nel lavoro di Tax [74] in cui viene dimostrato come i metodi matematici di un SVM possono essere adattati alla ricerca della miglior frontiera intorno ai dati di training. La frontiera più banale è rappresentabile con un centro $\mathbf{a}$ e un raggio R; per ricavare questi parametri viene nuovamente utilizzato un algoritmo di programmazione quadratica. Dopo aver introdotto delle variabili di errore simili alle $\xi_i \geq 0, i = 1, \cdots, l$ viste nella Sezione 2.6.1 è possibile ricavare quali sono i vettori di supporto della rappresentazione sferica dei dati e rappresentare tale frontiera $(\mathbf{a}, R)$ in termini dei soli vettori di supporto. Analogamente al caso di un SVM, la rapresentazione sferica dei dati è troppo rigida per essere utilizzabile in un ambiente reale. Con quello che è chiamato kernel trick, viene usata una funzione kernel con particolari proprietà per poter rappresentare frontiere arbitrariamente complesse come quelle in Figura 2.1.

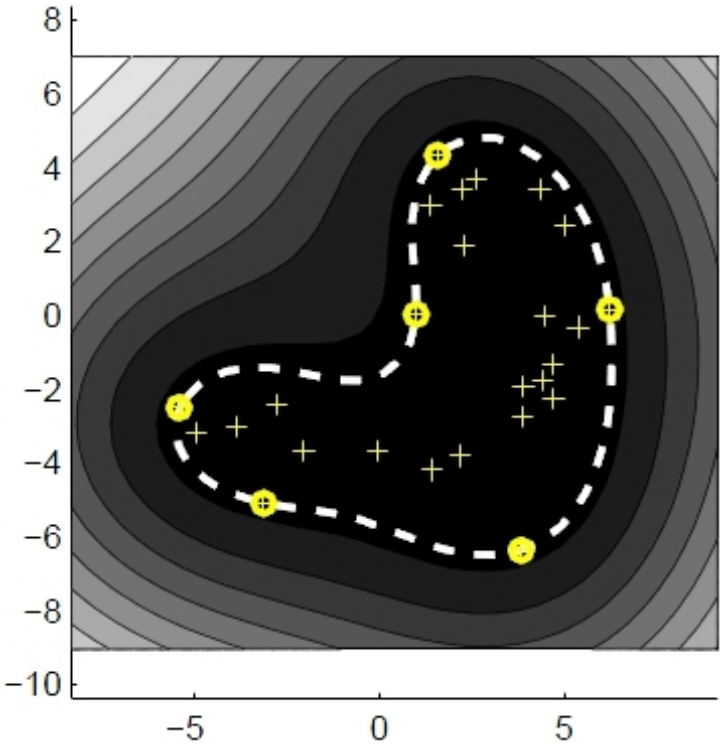


Figura 2.1: Esempio di frontiera non sferica costruita intorno ai dati di training

2.6.3 Classificatore SVM one-class

Con l'unione di SVM e SVDD è possibile costruire un classificatore one-class in grado di utilizzare esempi presi solo dalla classe target e in grado di determinare se un nuovo punto appartiene o meno a tale classe con un modello basato sulla frontiera costruita intorno ai dati di training. Ovviamente questa frontiera è necessariamente la più piccola possibile, perchè l'algoritmo di apprendimento utilizza solo i dati di una classe e non può avere informazioni sull'altro lato della frontiera stessa. In linea teorica è possibile (e vantaggioso) utilizzare anche esempi presi dalla classe degli outliers, spesso però è difficile fornire abbastanza esempi (uniformemente distribuiti) di questo tipo oppure è addirittura impossibile; per questo motivo tale approccio è molto utile soprattutto quando gli esempi sono disponibili solo ed esclusivamente per la classe target.

Il classificatore calcola il valore di decisione

$$f(\mathbf{z}, \boldsymbol{\alpha}, R) = K(\mathbf{z}, \mathbf{z}) - 2\sum_i \alpha_i K(\mathbf{z}, \mathbf{x}_i) + \sum_{i,j} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \leq R^2 \quad (2.10)$$

dove $\mathbf{z}$ è il punto incognito da classificare, $\boldsymbol{\alpha}$ è il vettore dei moltiplicatori di Lagrange, R è il raggio della frontiera, $K(\cdot, \cdot)$ è una funzione kernel e gli $\mathbf{x}_k$ sono i vettori di supporto. Per determinare se un nuovo punto è inlier o outlier, viene utilizzato un valore di soglia scelto in modo da ottimizzare il valore della probabilità a priori di avere outlier nel training set, ovvero della porzione di outlier che stiamo accettando. Infatti tale probabilità è uno dei due parametri del classificatore usato in questo lavoro di tesi (con un kernel RBF):

1. ν: massima frazione di outlier che possiamo a priori accettare nel nostro modello di classificazione

2. γ: coefficiente del kernel RBF. Tale valore è un indice della complessità della funzione kernel e definisce quindi la capacità di trasformare l'input in uno spazio di feature sempre più alto per poter separare le due classi linearmente

Il valore calcolato in (2.10) può essere inoltre utilizzato per estendere o ridurre la frontiera generata automaticamente dal classificatore; in sostanza è possibile determinare una soglia migliore di quella decisa dal classificatore per ottenere prestazioni potenzialmente migliori nel caso specifico in esame.

Un classificatore di questo tipo rende possibile l'approccio scelto per questa tesi: utilizziamo un serie di classificatori one-class per simulare il

comportamento di un classificatore a N classi ed avere la possibilità di ottenere come risposta anche la non appartenenza a nessuna della classi (ossia a nessun classificatore).

Capitolo 3

Item detection

L'obiettivo di questa tesi è quello di proporre un metodo per la classificazione automatica di tweet nei loro item corrispondenti. Ad esempio, se un utente sta *twittando* a riguardo di un certo film, il sistema deve classificare il tweet assegnandogli la classe di quel film (item). Più propriamente, questo processo può essere definito come **item detection**.

La preparazione dei dati per la fase di apprendimento parte dalla ricerca (attualmente effettuata in modo manuale) di pagine contenenti tweet che riguardano un certo item; con la notevole diffusione di Twitter su scala globale, non è difficile pensare di trovare pagine relative ad item come ad esempio dei film (§ 3.1.1). Per ogni item vengono poi scaricati e analizzati i tweet presenti nelle pagine corrispondenti per estrarne il testo ed eventualmente altri metadati (§ 3.1.2). Mediante una successiva analisi statistica dul contenuto dei vari tweet, vengono generate diversi tipi di feature (§ 4.2) che vengono poi utilizzate come vettori di training per un classificatore (§ 5.1.5). Questo procedimento viene eseguito per tutti gli item a disposizione e ogni classificatore sarà in grado di distinguere tweet provenienti dalla sua classe target (inliers) da quelli che non vi appartengono (outliers).

3.1 Dataset

Per eseguire gli esperimenti e validare i risultati, è stato utilizzato un dataset di tweet relativi ad un certo numero di item differenti. Tale dataset è stato costruito in modo da poter ottenere un numero di tweet per ogni item mediamente costante. Per ottenere risultati apprezzabili e piuttosto generici abbiamo selezionato solo tweet in lingua inglese, per cui la diffusione dei tweet è anche molto maggiore della media. L'estrazione dei tweet per il dataset è stata effettuata manualmente dalle pagine corrispondenti in quando

non è stato possibile trovare dei dataset pubblici di tweet, visto che Twitter li ha rimossi o non ne consente la diffusione. Per i pochissimi dataset di tweet disponibili, non esistono comunque le versioni annotate (nemmeno nel contesto dei film). I dataset testuali esistenti non sono utilizzabili in questo contesto perchè sono spesso documenti articolati e organizzati e non hanno la stessa natura dei commenti pubblicati in un microblog.

3.1.1 Fonti di informazione

Ogni profilo utente di Twitter è caratterizzato da uno specifico nome preceduto dal carattere @, quindi allo stesso modo le pagine relative ad un certo item (film) sono i profili utente ufficiali dell'item in questione. Tramite il meccanismo di follow è possibile per qualsiasi utente, ricevere gli aggiornamenti sui tweet pubblicati su un'altra pagina e anche effettuare retweet e risposte che poi possono essere a loro volta seguite. L'operazione di follow è già di per sè un grande indice di interesse dell'utente nei confronti di un certo item, è quindi plausibile che pubblichi un suo personale commento in risposta agli aggiornamenti delle sue pagine preferite. Nonostante non ci sia nessun vincolo sulla pertinenza che i tweet devono avere su un particolare pagina, l'assunzione forte alla base di questo lavoro è che i tweet contenuti nelle pagine ufficiali dei vari item siano realmente relativi al film in questione.

Sulla base di questa assunzione (che varrà comunque a livello statistico) abbiamo preso come item di apparteneza di ogni tweet, l'item relativo alla pagina su cui è comparso. Il dataset così creato, non è stato annotato a mano ma contiene un'annotazione implicita basata sulla precedente assunzione; questo ci consente di utilizzarlo anche per validare i risultati. É indubbiamente vero che un dataset fatto in questo modo potrà presentare errori nel training set e anche in fase di validazione: per far fronte a questa possibilità ci viene in aiuto ancora l'utilizzo di un SVM che è in grado di ottenere una funzione di classificazione anche tenendo in considerazione la possibilità di errori nel trainng set (§ 2.6).

La scelta dei profili ufficiali (pagine) è stata abbastanza ovvia, esistono molto spesso pagine Twitter relative a film. In questo lavoro sono state utilizzate le pagine relative ai film in Tabella 3.1.

3.1.2 API REST di Twitter

L'accesso e lo scaricamento dei tweet è stato effettuato tramite chiamate alle Twitter REST API versione 1.1. Esistono molti metodi sia GET che POST per accedere ai servizi forniti da Twitter ma, nonostante i contenuti

X-Men first class	Twilight	Immortals
Love and other drugs	The woman in black	The hunger games
Grabbers	Indie game	Heroine
Breaking dawn	Unconditional	Haynesville
Ecstasy	Moon point	Collapse
Lemon	The tunnel	Girltrash
The host	She wants me	Courageous
Looper	Bully	Super 8
Footloose	Transformers	Harry Potter
Hit and run	Titanic	Back to the future
Blue like jazz	Boy	Detropia
Paper heart	Star trek	Spartacus
Abduction	Spiderman	The vow
The last fall		

Tabella 3.1: Film utilizzati in questa tesi

screen_name	Il nome Twitter dell'utente (della pagina)
count	Specifica il numero di tweet restituiti per ogni richiesta
max_id	Utilizzato per la navigazione della timeline

Tabella 3.2: Parametri della chiamata REST all'API

dei tweet siano pubblici di natura, è necessario registrare un'applicazione sulla pagina D̈evelopersp̈er ottenere le chiavi e i tokens di accesso ai servizi non banali come può essere una ricerca libera (GET/search). La chiamata utilizzata in questo progetto è la:
`https://api.twitter.com/1.1/statuses/user_timeline.json`
che restituisce solo i tweet scritti dall'utente, in ordine cronologico inverso (timeline). Non è da confondere con la timeline che un qualsiasi utente di Twitter può vedere quando è loggato nel proprio account. Dei molti parametri utilizzati per questa chiamata, i principali sono mostrati nella Tabella 3.2.

La chiamata così costruita restituisce i risultati in formato JSON (JavaScript Object Notation), ad esempio:

```
{
    "created_at": "Wed Aug 29 17:12:58 +0000 2012",
    "id_str": "240859602684612608",
    "in_reply_to_user_id_str": null,
    "contributors": null,
```

```
    "text": "Introducing the Twitter Certified Products
         Program: https://t.co/MjJ8xAnT",
    "retweet_count": 121,
    "in_reply_to_status_id_str": null,
    "id": 240859602684612608,
    "geo": null,
    "retweeted": false,
    "possibly_sensitive": false,
    "in_reply_to_user_id": null,
    "place": null,
}
```


che sono poi parsati per salvare in modo più compatto le informazioni sul contenuto del tweet e sugli hashtag presenti.

3.1.3 Partizionamento e organizzazione del dataset

Prima di passare alla fase di generazione delle feature e di addestramento, è necessario organizzare il dataset suddividendolo in due partizioni:

- test set: una percentuale dei tweet di ogni item viene scelta casualmente e separata dal resto del dataset. La scelta casuale di questa frazione è dovuta al fatto che vogliamo creare un test set che includa tweet più o meno recenti, in grado di essere distribuiti uniformemente e non generare bias nella validazione
- training set: i tweet rimanenti saranno a loro volta privi sia di tweet recenti che di tweet più vecchi, avremo quindi anche un training set uniforme

Non tutti i tweet rimanenti sono però necessariamente utilizzati: in alcuni esperimenti il valore del volume del training set è un parametro e quindi è stata data la possibilità di modificarlo (Analisi di sensitività, Sezione 5.4.1).

3.2 Validazione

La fase di validazione è quella in cui vengono sottoposti alla batteria di classificatori un insieme si tweet che non sono mai stati visti dal sistema. Come precedentemente spiegato nella Sezione 3.1.3 all'inizio del procedimento abbiamo suddiviso il dataset in due parti. Il test set viene impiegato in questa fase per ottenere le prestazioni del sistema. I tweet contenuti in tutti i test set vengono quindi aggregati per formare un insieme di test complessivo

	Classe target	Classe outlier
Classe target	TP (True Positives)	FN (False Negatives)
Classe outlier	FP (False Positives)	TN (True Negatives)

Tabella 3.3: Confusion matrix

in cui ognuno dei classificatori troverà alcuni tweet appartenenti alla sua classe e altri che non vi appartengono. La nostra conoscenza a priori sull'appartenenza alle diverse classi ci permetti di stimare gli errori commessi dall'insieme di classificatori. Il test set complessivo così generato viene dato come input a tutti i classificatori: per ogni coppia tweet/item (t, i) otterremo quindi il valore di decisione come calcolato in (2.10) che corrisponde al grado di appartenenza del tweet t all'item i.

3.2.1 Metriche di valutazione

Per consentire un raffronto tra i diversi esperimenti eseguiti, sia con l'algoritmo di base che con l'algoritmo proposto, e analizzare i risultati in modo qualitativo e quantitativo, vengono utilizzate diverse metriche di valutazione. Tali metriche sono ricavate dalla matrice di confusione che a sua volta è una rappresentazione matriciale compressa dei risultati ottenuti da un classificatore. Nel nostro caso specifico, avendo solo una classe target per ogni classificatore, la matrice di confusione è una matrice 2X2 come quella in Tabella 3.3.

I valori nelle diverse celle rappresentano il numero di tweet della classe reale (nelle righe) e della classe predetta dal classificatore (nelle colonne). In questo modo possiamo sapere quanti tweet sono classificati in modo corretto o errato e come sono stati confusi quelli errati.

Le metriche che abbiamo utilizzato, ricavandole dalla matrice di confusione sono:

- Precision:

$$\frac{TP}{TP + FP} \tag{3.1}$$

  che è un indice della probabilità che un documento classificato come target lo sia veramente.

- Recall:

$$\frac{TP}{TP + FN} \tag{3.2}$$

che rappresenta la percentuale di documenti realmente della classe target, classificati come tali.

- Fallout:

$$\frac{FP}{FP + TN} \tag{3.3}$$

ovvero la percentuale di documenti classificati erroneamente come target.

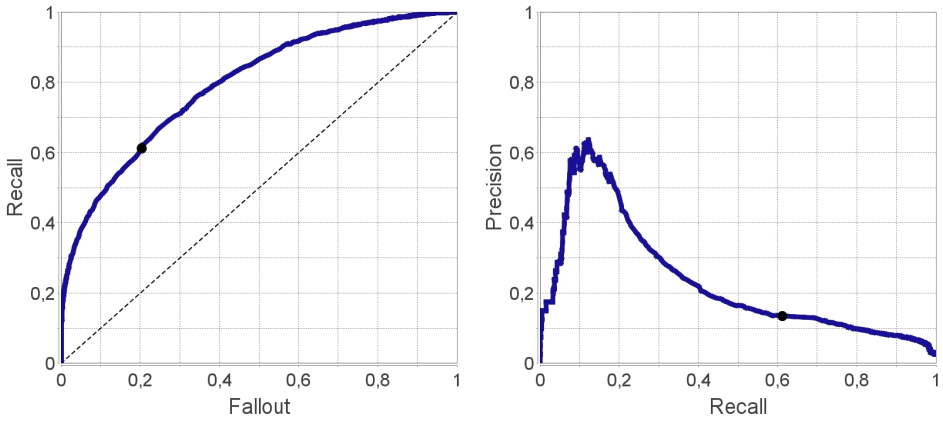
3.2.2 Curve di valutazione

I valori di precision e recall calcolati come nella Sezione 3.2.1 ci danno una visione istantanea di come funziona il singolo classificatore in una specifica condizione. Infatti i valori delle metriche sono soggetti a variazione in base ad un parametro di decisione (soglia) che il classificatore utilizza per discriminare le due classi. Nel nostro caso specifico, il classificatore addestrato nella fase di addestramento (§ 5.1.7) è costruito per distribuire i valoridi decisione sempre intorno al valore 0 e per utilizzare questo come soglia di decisione. La soglia ha come effetto quello di ridurre o aumentare in modo coerente le dimensioni della frontiera che è stata appresa. É quindi interessante valutare il comportamento di un classificatore al variare di questa soglia per poterne analizzare meglio le caratteristiche, utilizzare il risultato come termine di paragone e cercare il valore di soglia migliore (che non è necessariamente sempre quello deciso durante l'apprendimento).

La prima curva che possiamo calcolare è la **ROC (Receiver Operating Characteristic)** (Figura 3.1(a)) che ci permette di analizzare l'andamento congiunto delle metriche di recall e fallout al variare della soglia.

Questa curva ci permette di valutare il potere discriminativo del classificatore. Vediamo le sue caratteristiche di base:

- tutte le curve partono dal punto $(0, 0)$ e arrivano nel punto $(1, 1)$: questo perchè è banale ottenere valori di recall e fallout estremi, impostando le soglie minima e massima. In questi non vi è nessuna discriminazione tra le classi
- il classificatore perfetto è quello che raggiunge il punto $(0, 1)$
- la bisettrice del grafico rappresenta la curva ROC di un ipotetico classificatore casuale a due classi. Questa retta è il minimo valore che possiamo tollerare



(a) ROC (b) Precision/Recall

Figura 3.1: Esempi di curve di valutazione

- una generica curva ROC si pone tra la bisettrice e la linea del classificatore ideale e l'area sottesa al grafico (Area Under Curve) è una misura del suo potere discriminativo.
- anche la convessità della curva stessa è un indice del b̈uonc̈omportamento del classificatore
- è possibile scegliere il punto migliore sulla curva in base a diverse politiche e utilizzare la soglia corrispondente per far lavorare il classificatore in quel punto

La seconda curva che usiamo per l'analisi è la curva di **Precision/Recall (PR)** (Figura 3.1(b) in cui viene visualizzato l'andamento della coppia precision/recall; sempre variando il valore della soglia tra i due estremi. In questo caso non ci sono caratteristiche comuni per l'andamento di tale curva ma possiamo in ogni caso elencare alcune regole per la sua analisi:

- il valore della recall varierà obbligatoriamente tra 0 e 1, come abbiamo già spiegato nella sua definizione
- anche la precision è vincolata ad essere tra 0 e 1 perchè è una percentuale
- è possibile visualizzare immediatamente quale sia il valore massimo di precision e come esso varia al variare della soglia
- la scelta del punto di lavoro è subordinata ad un valore alto di precision ma congiuntamente alla recall corrispondente. Anche se auspicabile,

non è necessario raggiungere una elevata precision in corrispondenza di un'alta recall, abbiamo la possibilità di preferire una precision alta anche in presenza di una recall non troppo elevata. Inoltre è possibile imporre dei vincoli al valore di uno delle due metriche e sapere a priori quali saranno i valori corrispondenti assunti dall'altra metrica

Entrambe le curve sono calcolate per ogni singolo classificatore per poterne osservare il comportamento. Per analizzare e confrontare il comportamento del sistema nel suo complesso al variare di feature e altri parametri abbiamo calcolato le curve ROC e PR globali utilizzando semplicemente la media aritmetica (macro-avarage) dei valori di precision, recall e fallout nei corrispondenti valori di soglia. L'utilizzo della semplice media aritmetica è dovuto al fatto che per come è stato costruito il dataset, non ci sono grosse differenze per il training dei singoli classificatori. Queste due curve globali sono quelle di riferimento per il confronto tra esperimenti diversi e ci permettono di valutare il funzionamento medio del del sistema di classificazione, sia esso quello banale che il nostro. É da tenere in considerazione che per questa fase i valori delle metriche sono calcolati accettando il fatto che più di un classificatore possa rispondere in modo positivo per lo stesso tweet.

3.3 Dettagli tecnici e implementativi

L'intero progetto è stato sviluppato in Java con l'utilizzo dell'IDE Eclipse. Oltre al codice Java prodotto, sono state utilizzate alcune librerie di supporto:

- Apache HttpComponents[1] v4.2.1: per la gestione delle richieste HTTP e l'utilizzo più standard degli oggetti richiesta e risposta
- Signpost[2] v1.2.1.2: per gestire in modo automatico la firma dei messaggi HTTP necessaria per le richieste REST alle API di Twitter con il protocollo OAuth. Integrata con Apache HttpComponents
- Java JSON[3]: per il parsing degli oggetti JSON restituiti dalle chiamate HTTP
- OpenNLP[4] v1.5.2: per l'estrazione dei token dai documenti. É stato utilizzato un modello precedentemente addestrato per la lingua inglese,

[1] http://hc.apache.org/
[2] http://code.google.com/p/oauth-signpost/
[3] http://www.json.org/java/
[4] http://opennlp.apache.org/

disponibile a questo indirizzo:
`http://opennlp.sourceforge.net/models-1.5/`

- LibSVM[5] v3.13: libreria in versione Java per l'addestramento di classificatori SVM; contiene il codice per effettuare la crossvalidazione
- JFreeChart[6] v1.0.14: per la creazione di grafici, sempre in ambiente Java

Inoltre sono stati utilizzati anche:

- una lista di stopword, disponibile al link
`http://www.ranks.nl/resources/stopwords.html`
- una implementazione dello stemmer di Porter, disponibile qui
`http://tartarus.org/martin/PorterStemmer/java.txt`

Il progetto è stato sviluppato utilizzando 40 item (film) ognuno con la relativa pagina ufficiale. La scelta di 40 item è stata pilotata dalla necessità di ottenere almeno 800 tweet per item: anche se oggi non è facile trovare molte pagine relative ad item con molti tweet, visto che Twitter è un social network ancora in forte espansione saranno sempre di più le pagine molto popolari riguardanti i film. Una prova di questo è che nel nostro insieme di item ne esistono alcuni relativi a film non ancora disponibili; a sottolineare come l'utilizzo di Twitter possa in un futuro molto prossimo interessare non solo l'analisi critica dei tweet da parte degli utenti ma anche uno strumento per anticiparne i gusti e le preferenze. Sono stati scaricati i primi 800 tweet (in ordine cronologico inverso) dei 40 item di nostro interesse (in lingua inglese, per avere una probabilità alta di ottenere molti tweet) per un totale di circa 32000 tweet equamente distribuiti. Questi tweet sono stati suddivisi in questo modo: circa il 25% dei tweet è stato separato in modo casuale dall'insieme di tweet di ciascun item per costituirne il test set, mentre del restante sono stati utilizzati un numero di tweet variabile (in base ai test da effettuare) presi in ordine cronologico inverso (come restituiti dalla chiamata al servizio di Twitter.

Prima della fase di addestramento il training set di ciascun item è stato suddiviso in 5 parti per ricavare il valore di crossvalidazione utilizzando ciclicamente ogni parte come set di validazione. La ricerca dei parametri migliori è stata effettuata in una sottoporzione dello spazio dei parametri che varia:

[5] http://www.csie.ntu.edu.tw/ cjlin/libsvm/
[6] http://www.jfree.org/jfreechart/

- per ν da 0.1 a 0.2 (compresi) con un passo di 0.05 perchè i valori ottimi per tutti gli item si trovano all'interno di questo range molto piccolo
- per γ da 0.0001 a 1000 (compresi), aumentando ad ogni iterazione l'ordine di grandezza

Capitolo 4

Algoritmo di base

Il primo passo dell'analisi è quello di verificare le prestazioni di un algoritmo di classificazione basato su delle query utilizzando alcuni dati noti a priori sugli item tra tutti i dati disponibili. L'informazione più basilare di tutti è il titolo del film: abbiamo quindi provato alcuni approcci per capire quale fosse il miglior algoritmo di partenza con il quale confrontarci.

4.1 Preprocessing

Prima di poter analizzare le feature per ogni tweet è necessario definire un vettore delle feature. Questo vettore rappresenterà il tweet in uno spazio in cui le dimensioni sono le feature stesse. Ogni tweet avrà quindi un valore per ognuna delle feature che compone il vettore e potrà essere rappresentato come un punto nello spazio delle feature; in questo modo i documenti possono essere posti ad una distanza che ne rappresenta la similarità. I diversi tipi di feature sono tutti basati sul contenuto del tweet e quindi sui termini che compaiono al suo interno. Prima di poter effettuare un conteggio sulla base del quale ricavare un valore statistico da attribuire ad ogni termine, i tweet vengono puliti e normalizzati tramite i seguenti passaggi:

1. **Tokenizzazione**: ogni tweet viene suddiviso in token con l'utilizzo della libreria OpenNLP. Questo passaggio è necessario per decidere se e dove dividere il testo in presenza della punteggiatura. Nell'esempio abbiamo separato i token con un trattino:

   What - do - you - love - most - about - Twilight - #BreakingDawnPart2 - ? - RT - if - you're - seeing - it - again - this - weekend! - http://bit.ly/BD2-tix

2. **Pulizia**: vengono eliminati i token (in rosso) che non corrispondono a termini validi tramite un'espressione regolare. Tra questi ci sono valori numerici, punteggiatura, URL, lettere ripetute e singole e sequenze di simboli particolari. Vengono inoltre eliminati i termini frequenti come R̈T̈(retweet) e ḧttp̈(frequentemente isolato come token singolo):

   What - do - you - love - most - about - Twilight - #BreakingDawnPart2 - ? - RT - if - you're - seeing - it - again - this - weekend! - http://bit.ly/BD2-tix

3. **Forme flesse**: nel mondo di Twitter il limite dei 140 caratteri costringe gli utenti ad utilizzare le forme flesse previste dalla lingua. In questo passaggio le contrazioni vengono normalizzate nella loro forma completa:

   What - do - you - love - most - about- Twilight - #BreakingDawnPart2 - if - you're (you are) - seeing - it - again - this - weekend!

4. **Caratteri speciali**: alcuni token possono essere composti da una parola più un simbolo iniziale o finale perchè riguardano ad esempio una domanda o un'esclamazione; in questo caso la parola viene ripulita dai simboli speciali per poter essere utilizzata come un termine normale:

   What - do - you - love - most - about - Twilight - #BreakingDawnPart2 - if - you - are - seeing - it - again - this - weekend! (weekend)

5. **Stemming**: tutti i termini che hanno raggiunto questo punto dell'algoritmo sono parole realmente esistenti nel vocabolario, ma sono ancora differenti tra loro per quanto riguarda il genere, il numero, il tempo verbale o la declinazione. Il metodo per accomunare tutti i termini che appartengono alla stessa famiglia è quello di associare al gruppo una stringa di testo rappresentativa: lo stem. L'algoritmo è composto da più fasi e si preoccupa di eliminare i caratteri che rappresentano una particolare forma di ogni termine, riducentoli così alla sola radice:

   What - do - you - lov - most - about - Twilight - #BreakingDawnPart2 - if - you - ar - see - it - again - thi - weekend

6. **Stopword**: in tutte le lingue esistono parole che vengono utilizzate con frequenza molto maggiore di altre. Queste parole, se inserite nel sistema di classificazione, rovinano le prestazioni perchè sono uniformemente distribuite in tutti i documenti e non caratterizzano nessuna categoria o topic, e come precedentemente detto, sono molto frequenti ottenendo dei valori molto alti per le feature corrispondenti. Per ovviare a questo problema abbiamo scelto un gruppo di 671 parole molto comuni nella lingua inglese e le abbiamo eliminate dalla lista di termini presenti nei tweet (in rosso):

   What - do - you - lov - most - about - Twilight - #BreakingDawnPart2 - if - you - ar - see - it - again - thi - weekend

Dopo tutti questi passaggi, i termini rimanenti sono pronti per essere contati per ricavarne una frequenza relativa e successivamente un valore da associare ad ogni specifico termine. Questa analisi statistica è compiuta su tutti i tweet presenti nei training set di tutti gli item.

4.2 Feature

Dopo un'analisi dei tweet e con riferimento a quanto viene solitamente utilizzato nell'analisi del testo lungo (§ 2.3.1), i tipi di feature che abbiamo scelto di utilizzare sono: unigrammi, bigrammi, hashtag e titolo. Ciascuna feature rappresenterà una dimensione nel vettore rappresentante i tweet.

Unigrammi: in questo caso vengono analizzate le frequenze dei singoli termini all'interno dei documenti. É opportuno sottolineare che in questo tipo di feature talvolta ricadono anche alcuni hashtag il cui simbolo caratteristico #̈viene separato in fase di tokenizzazione; nel caso degli unigrammi però il peso relativo delle feature in forma IDF è notevolmente ridotto in quando un termine utilizzato in un hashtag può essere utilizzato anche come termine testuale. Dopo l'analisi statistica, vengono eliminati i termini che compaiono una volta sola e viene così costruito il vettore di feature per gli unigrammi composto da un valore (IDF o binario) per ciascun termine all'interno dei tweet di training di tutti i 40 items). Nell'esempio evidenziamo in rosso alcuni dei possibili unigrammi che vengono estratti:

What - do - you - love - most - about - Twilight - #BreakingDawnPart2 - ? - RT - if - you're - seeing - it - again - this - weekend! - http://bit.ly/BD2-tix

Bigrammi: questo tipo di feature cerca di cogliere un aspetto più avanzato dei singoli termini, ovvero le occorrenze frequenti di coppie adiacenti di termini. Anche in questo caso vengono eliminati i bigrammi che compaiono una sola volta e viene costruito il vettore di feature. É da sottolineare come in questo caso il processo di stopwording sia differente: non è più possibile eliminare le stopword prima di cercare i bigrammi all'interno del testo, in questo modo perderemmo alcuni digrammi che contengono magari una sola stopword. Quindi vengono prima ricercati tutti i bigrammi nel testo e solo successivamente eliminati quelli che contengo 2 termini entrambi appartenenti alle stopword (come äs iföppure ÿou are¨). Nell'esempio vengono visualizzati in rosso alcuni dei bigrammi presenti:

What - do - you - love - most - about Twilight - #BreakingDawnPart2 - ? - RT - if - you're - seeing - it - again - this weekend! - http://bit.ly/BD2-tix

Hashtag: gli hashtag (in rosso nell'esempio) sono delle parole che compaiono nel testo precedute dal simbolo #¨e in Twitter hanno un significato particolare: sono infatti delle chiavi per la ricerca di tweet. Vengono utilizzato in molti casi per sottolineare un concetto o dargli più enfasi e sono spesso strettamente correlati con il topic in questione; inoltre gli hashtag sono normalmente rappresentati nei tweet come link alla ricerca ci appartengono, ciò li rende particolarmente potenti e diffusi perchè è possibile avviare la ricerca con un singolo click. In questo caso la tokenizzazione viene controllata in modo da non separare (e quindi perdere) alcuni degli hashtag nel testo e successivamente si applica ancora lo stesso criterio per la creazione del vettore di feature utilizzato nel caso degli unigrammi. Come giò accennato, gli hashtag possono anche essere composte da parole di uso comune, in questo caso però la loro frequenza è minore e di conseguenza il loro valore IDF più alto, rendendoli più discriminativi.

What - do - you - love - most - about - Twilight - #BreakingDawnPart2 - ? - RT - if - you're - seeing - it - again - this - weekend! - http://bit.ly/BD2-tix

Titolo: questo tipo di feature sono il tentativo di applicare l'algoritmo di base al caso dei classificatori del nostro sistema. Viene creata una feature per ogni item nel sistema, quindi otterremo un vettore di feature con la stessa cardinalità degli item. Il valore di una feature per un tweet t rispetto all'item i è quindi la percentuale di parole trovate nel tweet t relativa al titolo del film dell'item i:

$$f(t,i) = \frac{|(t \mid t \in \text{Titolo}(i))|}{|\text{Titolo}(i)|} \tag{4.1}$$

dove Titolo (i) è l'insieme dei termini contenuti in nel titolo dell'item i. Nell'esempio seguente possiamo vedere in rosso il titolo trovato all'interno del tweet (in questo caso una parola sola):

What - do - you - love - most - about - Twilight - #BreakingDawnPart2 - ? - RT - if - you're - seeing - it - again - this - weekend! - http://bit.ly/BD2-tix

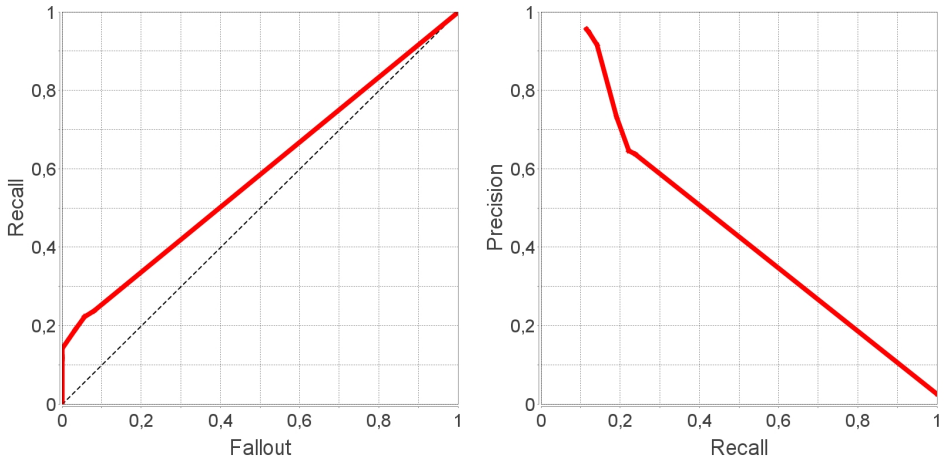
4.3 Algoritmo

Delle feature studiate, abbiamo quindi utilizzato solo il titolo in una versione adattata per un approccio basato sul keyword matching. Abbiamo individuato le seguenti 4 varianti: (i) Exact matching, (ii) Token matching, (iii) Free matching e (iv) Advanced matching.

Exact matching. In questo caso abbiamo cercato (e classificato come inliers) i tilm che contenevano **esattamente** il titolo esatto del film in questione. Il testo è stato normalizzato per non tenere in considerazione i caratteri maiuscoli. In questa prima versione otteniamo una precision del 96%, una recall dell'11% e un fallout infinitesimo. Questi risultati sono interpretabili in modo molto diretto: è quasi ovvio anche a priori che un tweet che contiene il titolo esatto di un film, sia correlato a tale film. Da queste considerazioni si può spiegare una precision così alta e un fallout così basso ma a discapito di una recall molto bassa, dovuta al fatto che non è così frequente che il titolo esatto sia presente nel tweet. Questo comportamento è facilmente spiegabile: in un sistema di microbloggin non è necessario quasi mai esplicitare l'argomento di discussione, perchè i tweet che vengono pubblicati sono di fatto parti di una conversazione.

Token matching. Il passo successivo è stato quello di **spezzare il titolo** nelle sue parole e cercare all'interno dei tweet almeno uno di questi termini. Il problema di questo approccio è quello di essere molto dipendente dal titolo che stiamo analizzando.

Da questa versione in avanti, è concepita la possibilità di applicare una soglia alla percentuale di parole del titolo presenti nel tweet. Come si vede dalla Figura 4.1, le prestazioni di precision per la soglia massima corrispondono alle prestazioni della prima versione dell'algoritmo, in quanto è quasi



(a) ROC

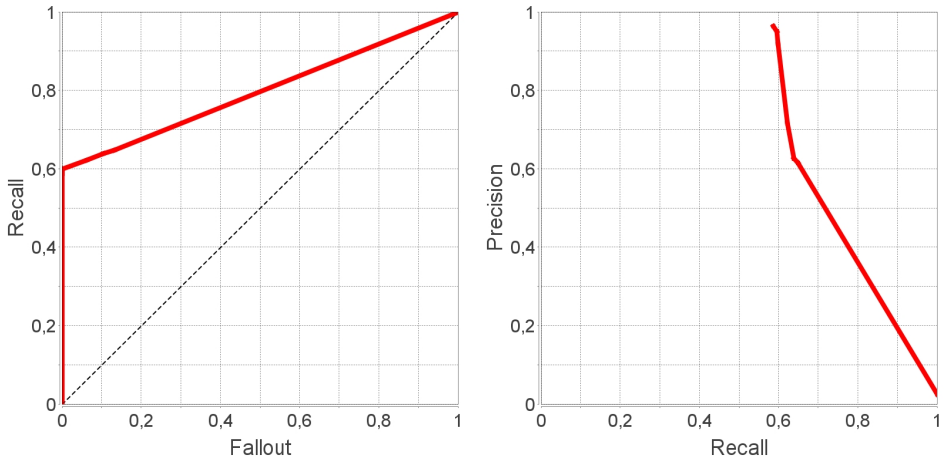
(b) Precision/Recall

Figura 4.1: Algoritmo di base, versione Token matching

impossibile trovare tutte le parole del titolo di un film in un ordine diverso rispetto a come sono nel titolo stesso. Si può notare un piccolo potere discriminativo nella ROC che è la motivazione principale per cui la precision scende velocemente all'aumentare della recall. Questo test ci permette principalmente di analizzare come varia la precision al variare di questa soglia, senza andare oltre le prestazioni migliori della versione Exact matching.

Free matching. La differenza con il punto precedente è che in questo caso vogliamo cercare i termini che compongono il titolo anche **all'interno** di altre parole. Questo metodo ci permette di ricavare dei match anche prendendo in considerazione gli hashtag e le mention, nonchè alcuni errori di composizione del tweet.

É in questo esperimento che si ha il maggior incremento di prestazioni; in Figura 4.2 possiamo notare dalla ROC come il potere discriminativo si sia molto alzato, ma soprattutto dalla curva di Precision/Recall come, mantenendo sempre una precision massima del 96%, abbiamo ottenuto una recall vicina al 58% che corrisponde ad una fallout quasi nulla. L'incremento così forte della recall è dovuto al fatto che, non limitandoci più alla ricerca dei termini come parole intere, possiamo cercare alcune parole del titolo (ma anche il titolo intero ad esempio senza spazi) come hashtag, mentions o alcune abbreviazioni. Visto che l'utilizzo di tali metadati è molto frequente in Twitter, possiamo ottenere questo forte incremento della recall. Ovviamente la soglia sulla percentuale delle parole del titolo contenute nel tweet deve essere mantenuta alta (almeno sopra 0.75) per ottenere buoni risultati, in



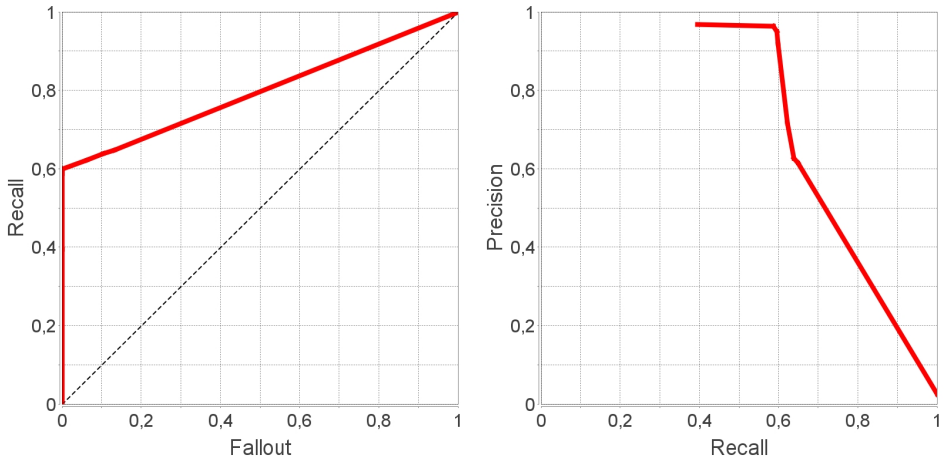
(a) ROC (b) Precision/Recall

Figura 4.2: Algoritmo di base, versione Free matching

caso contrario andremmo incontro ad un calo della precision (come si vede nel grafico) perchè andremmo a classificare alcuni tweet per una classe anche in presenza di poche parole, poco significative o addirittura stopword. Tale problema sarà sempre più grande quando ci saranno item con titoli composti da molte parole.

Advanced matching. Quest'ultima opzione unisce l'approccio Exact matching e il Free matching per ottimizzare le prestazioni. In una prima fase viene effettuata una ricerca del titolo completo ma in modo libero (con la possibilità di fare un matching anche su titoli circondati da caratteri speciali). Se non vi è stato un matching immediato, si cercano i singoli termini che compongono il titolo stesso con il metodo Free matcihng.

Possiamo notare dalla Figura 4.3 come le prestazioni siano rimaste pressochè invariate rispetto alla versione Free matching. Infatti questa versione include il controllo preventivo del titolo nel tweet (come nella versione 1) che però è già contemplato dalla ricerca libera dei termini. Il motivo per cui viene usata questa quarta versione è che possiamo dare un peso specifico diverso a quei match che riguardano il titolo nel suo ordine corretto. Da notare come il grafico di Precision/Recall mostri un nuovo punto con precisione migliore (circa 97%) ma recall intorno al 39%. Questo punto è quello in cui viene utilizzato solo il titolo nel suo ordine corretto; a differenza del caso 1 però possiamo cercare il titolo in modo libero e aumentare la recall includendo nei match possibili quei casi in cui il titolo è delimitato da qualche errore o carattere speciale. Concludendo, questa prima analisi degli



(a) ROC

(b) Precision/Recall

Figura 4.3: Algoritmo di base, versione Advanced matching

algoritmi basati sul keyword matching ci ha fornito un algoritmo di base con prestazioni sorprendenti rispetto alle aspettative.

Capitolo 5

Algoritmo SVM

Nel Capitolo 4 abbiamo valutato le performance dell'algoritmo base, basato solo sul titolo dell'item. Proseguiamo ora lo studio con una metodologia più avanzata. Tale metodologia prevede una prima fase di analisi dei training set relativi ad ogni item per costruire un vettore di feature più o meno articolato (§ 5.1.3) per poi utilizzarlo come base per generare le feature relative ad ogni tweet. I vettori di features (§ 4.2) presenti nei training set di ogni item vengono utilizzati in un classificatore one-class SVM per cercare i parametri migliori di funzionamento (§ 5.1.6) e per effettuare il reale addestramento (§ 5.1.7). Il sistema finale (Figura 5.1) conterrà quindi un classificatore one-class SVM per ciascun item, ognuno dei quali sarà in grado di separare i tweet in quelli appartenenti a tale classe e negli outlier. Infine i test set di ogni item vengono utilizzati insieme come test set complessivo per tutti gli item. Su questo gruppo di tweet viene effettuata la predizione da parte di tutti i classificatori e vengono calcolate le matrici di confusione relative (§ 3.2). Per visualizzare i risultati e renderli chiaramente confrontabili abbiamo disegnato le curve ROC (Receiver Operating Characteristic) e Precision/Recall sia per i singoli classificatori che come medie di tutto il sistema (§ 3.2.2).

5.1 Algoritmo

Il sistema nel suo complesso sarà quindi composto da una batteria di classificatori one-class che lavoreranno in parallelo ognuno per produrre il suo risultato locale, simulando quindi un sistema multiclasse anche in grado di assegnare un nuovo tweet a nessuna delle classi (item) esistenti. Per ogni nuovo tweet in ingresso vengono quindi generate le relative features e vengono date in ingresso ad ognuno dei classificatori. Un approccio di questo tipo

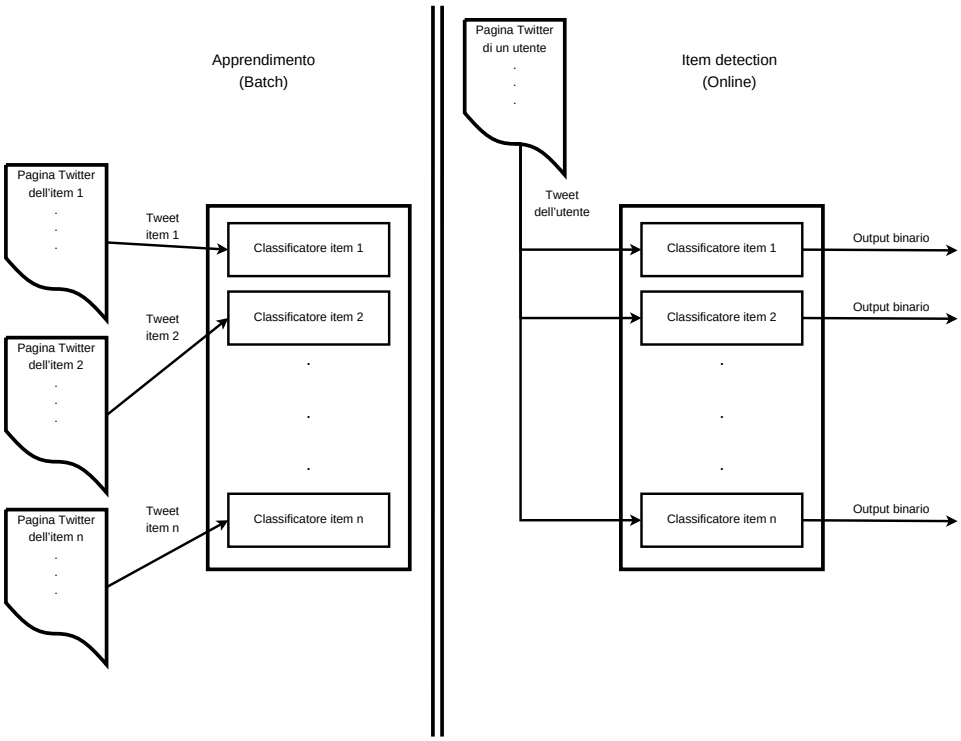


Figura 5.1: Struttura dell'algoritmo SVM

offre diversi vantaggi per una futura applicazione integrata con un sistema di raccomandazione:

1. contrariamente ad un sistema basato su un classificatore multiclasse, è possibile utilizzare un numero di classi (e quindi classificatori one-class) molto elevato

2. il training iniziale viene eseguito per tutti gli item disponibili, ma l'aggiunta di un nuovo item nel sistema non richiede quasi mai un nuovo training di tutta la batteria di classificatori

3. è possibile ottenere anche il risultato di non appartenenza ad alcuna classe (non disponibile con un classificatore multiclasse)

4. le diverse risposte locali dei diversi classificatori ci permettono anche di inferire un certo grado di similarità tra item: in un sistema completo di raccomandazione sarebbe plausibile che un tweet possa anche appartenere a più classi diverse simili tra loro. Il compito di scegliere quale sia l'unica classe reale di appartenenza di un tweet è un compito non banale perfino per un essere umano

Nelle prossime sezioni vedremo quali sono i passi necessari per effettuare l'addestramento della batteria diclassificatori. Partendo dall'analisi e pulizia delle feature (§ 5.1.1), al calcolo del corrispondente valore IDF (§ 5.1.2) fino alla definizione del vettore di feature da utilizzare (§ 5.1.3). Successivamente vengono generate le feature per tutti i tweet di training e i test (§ 5.1.4) per poi essere usate per addestrare i classificatori (§ 5.1.5). Infine sarà possibile utilizzare il sistema per valutarne le prestazioni 3.2.

5.1.1 Eliminazione feature inutili

Dopo il conteggio delle occorrenze di tutti i termini otteniamo una distribuzione dei termini sui diversi item, scoprendo quali sono i termini più o meno utilizzati in ogni item. Come già visto per le stopword, in questa fase possiamo eliminare invece i termini che compaiono solo una volta, perchè è molto probabile che siano parole rare, piuttosto che termini contenenti errori di grammatica non corretti ai passi precedenti. Tali termini non saranno quindi presi in considerazione dal classificatore perchè quasi mai presenti e non aggiungono potere discriminativo, inoltre questo passo serve anche per ridurre drasticamente il numero delle dimensioni e ottenere vettori di feature più piccoli. Nonostante le considerazioni teoriche portino a pensare con

discreta certezza che questa fase sia necessaria e utile a migliorare le prestazioni del sistema, questa variabile è stata analizzata nella fase sperimentale (§ 5.2.1) con risultati sorprendenti, che ci ribadiscono come l'ambiente del microblog sia particolare.

5.1.2 IDF

La generazione delle feature (§ 5.1.4) può essere effettuata in diversi modi; come prima scelta si potrebbe pensare di utilizzare un approccio in cui si valuta la presenza di un termine in un tweet senza prendere in considerazione la sua frequenza assoluta e relativa: in questo caso le feature avrebbero valori binari. I problemi di questo tipo di possono essere:

- la feature non conterrebbe informazioni riguardo la frequenza di un termine, distribuendo uniformemente la probabilità di occorrenza dei termini; in questo modo potrebbe ridursi la capacità di discriminare nel modo corretto
- non è possibile assegnare valori bassi alle parole più diffuse (come se fossero delle stopword relative al contesto) e quindi diminuirne il peso in fase di addestramento

Nell'analisi del testo classica, per rappresentare i documenti nello spazio delle feature con quello che viene chiamato Vector Space Model, si possono utilizzare anche diversi valori estratti facendo statistiche sui termini. Il più diffuso è sicuramente il TF-IDF (Salton and Buckley 5):

$$\mathrm{tfidf}\,(t, d, D) = \mathrm{tf}\,(t, d) \cdot \mathrm{idf}\,(t, D) \tag{5.1}$$

con

$$\mathrm{tf}\,(t, d) = \frac{\mathrm{f}\,(t, d)}{\max\{\mathrm{f}\,(w, d) : w \in d\}} \tag{5.2}$$

$$\mathrm{idf}\,(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{5.3}$$

dove t sono i termini, d i documenti, $\mathrm{f}\,(t, d)$ è la frequenza del termine t nel documento d, $|D|$ è la cardinalità dell'insieme dei documenti e $|\{d \in D : t \in d\}|$ è il numero di documenti in cui il compare il termine t. Il valore tf tiene in considerazione la frequenza dei vari termini nei documenti, mentre il valore idf è una misura della popolarità di un termine in tutto l'insieme di documenti, che quindi deve essere sottopesato perchè le frequenze

di quel termine sono dovute solo alla sua popolarità. Gli esempi classici di termini molto frequenti nel contesto dei film sono ad esempio "movie" oppure "DVD".

Nell'amiente di Twitter però, il testo a disposizione per ogni documento (tweet) è di soli 140 caratteri, quindi è quasi impossibile trovare dei termini ripetuti all'interno dello stesso docuemnto. Il valore di tf utilizzato in questo lavoro è quindi impostato a 1: il valore delle features sarà quindi dipendente solo da idf (5.3). In questo modo possiamo avere feature non binarie che tengono in considerazione la possibilità sottopesare i termini troppo frequenti. Visto che il valore di idf dipende solo dal termine t e non dal documento d, possiamo dire che i valori assunti dalle feature relative allo stesso termine nei vari tweet sono 0 oppure $\text{idf}(t, D)$.

Il logaritmo utilizzato nel calcolo dell'idf (5.3) non distribuisce i valori in modo inversamente proporzionale, per ovviare al problema per cui un termine che compare una volta sola abbia valore idf uguale a n e quello che compare 2 volte abbia già valore $n/2$. Per ottimizzare le prestazioni e l'accuratezza dei classificatori one-class SVM (§ 5.1.5), è consigliabile normalizzare i vettori di feature per avere solo valori compresi tra 0 e 1; quindi il risultato del calcolo idf (che varia in un range finito) viene in una fase finale normalizzato tra questi due estremi.

5.1.3 Generazione del vettore di features

Il vettore di feature contiene le informazioni sul valore idf e la struttura e ordinamento delle feature che vengono utilizzate. Per ogni tipo di feature, viene calcolato un vettore p̈arzialec̈he poi è normalizzato; la normalizzazzione avviene separatamente per ogni vettore parziale, in modo che le feature abbiano un peso relativo al loro gruppo, che può avere caratteristiche e distribuizioni molto differenti dagli altri. In questo modo è possibile tenere in considerazione per esempio che un termine singolo e lo stesso termine utilizzato come hashtag abbiano pesi differenti.

Dopo la generazione dei vettori parziali, questi conterrano già solo valori compresi nell'intervallo $[0, 1]$ e potranno essere uniti in un unico vettore di feature, quello finale. In questo modo è possibile combinare più tipi di feature tra loro per verificarne l'accuratezza.

La disposizione delle feature all'interno del vettore finale non influisce sul comportamento e le prestazioni del classificatore, visto che la distribuzione dei termini all'interno dei vari item non è nota a priori e che il classificatore riesce a separare linearmente le classi con il kernel trick. Se fosse nota qualche informazione a priori si potrebbe costruire un vettore di feature or-

ganizzato in modo da semplificare (non migliorare in termini di accuratezza) il lavoro del classificatore, ma sarebbe anche piuttosto inutile dato che saremmo già a conoscenza di molte informazioni per classificare i documenti nei vari item.

5.1.4 Generazione delle features

La fase di generazione delle feature vere e proprie è quella in cui, seguendo l'ordinamento del vettore di feature, viene generata un'istanza di questo vettore per ogni tweet. Questo vettore rappresenta ogni particolare tweet nello spazio delle feature e contiene i valori delle singole feature. Questa trasformazione ci consente di utilizzare i documenti come vettori in uno spazio e quindi di utilizzare un classificatore per separare in classi la popolazione di documenti. Chiaramente il classificatore avrà un'accuratezza migliore quando le feature sono sensate e rappresentano davvero il contenuto delle caratteristiche che vogliamo discriminare con tale metodo.

Nel nostro caso specifico, le feature rappresentano un valore di frequenza dei termini all'interno del documento stesso, con particolare attenzione alla frequenza assoluta e relativa (§ 5.1.2); adottiamo quindi un approccio semantico basato sulla linguistica pr cercare di classificare i tweet nei singoli item. Come già accennato il valore delle feature per un termine t può essere 0 oppure $\mathrm{idf}(t, D)$ nel caso di feature reali, oppure 0 o 1 nel caso di feature binarie.

5.1.5 Classificazione

Il processo di classificazione è composto da due fasi separate: una prima fase di Cross validazione per la ricerca dei parametri migliori di lavoro e la fase vera e propria di addestramento in cui viene costruito il modello in grado di effettuare le predizioni. Questo procedimento è effettuato per ogni item, creando una batteria di classificatori one-class. Nelle prossime sezioni vedremo nel dettaglio queste due fasi.

5.1.6 Cross validazione

La fase di Cross validazione è necessaria per cercare i migliori parametri per ogni classificatore:

- ν: massima frazione di outlier che possiamo a priori accettare nel nostro modello di classificazione. Questo parametro fa in modo che il classificatore non debba necessariamente cercare la frontiera che separi

esattamente i punti in inlier e outlier, ma possa accettare soluzioni considerate migliori che però includono degli outliers nel training set. Questo parametro può variare in $(0, 1]$.

- γ: coefficiente del kernel RBF. Tale valore è un indice della complessità della funzione kernel e definisce quindi la capacità di trasformare l'input in uno spazio di feature sempre più alto per poter separare le due classi linearmente. Tale parametro consente di far fronte a distribuzioni molto complesse in uno spazio multidimensionale a cardinalità molto elevata; ovviamente utilizzando funzioni sempre più complesse, otterremo modelli complessi e composti da molti vettori di supporto, necessari per la corretta predizione. I valori possibili per questo parametro sono

Per trovare la coppia di parametri migliore (o una delle migliori) viene effettuata una ricerca nello spazio bidimensionale dei due parametri e per ogni coppia di valori viene estratto un valore di accuratezza con l'algoritmo di Cross validazione:

$$\frac{TP}{TP + FN} \tag{5.4}$$

dove $TP + FN$ corrisponde al numero totale di campioni, visto che il training è composto solo da tweet della classe target. I parametri corrispondenti all'accuratezza migliore, sono scelti come valori ottimali. Nel nostro caso specifico, per non appesantire il processo, sono stati effettuati dei test sullo spazio dei parametri per cercare una porzione ottima (o sub-ottima) nella quale cercare la coppia candidata con una granularità più piccola. Ulteriori test ci hanno mostrato che la porzione ottima rimane pressochè la stessa se rimaniano nel campo applicativo dell'item detection in Twitter applicata ai film.

5.1.7 Addestramento

La coppia migliore di parametri (ν, γ) completa quindi lo spettro dei parametri necessari al classificatore per addestrarsi e costruire un modello per le predizioni. Infatti per motivi tecnici (§ 3.3) esistono altri parametri necessari alle librerie per funzionare correttamente. La fase di addestramento è preceduta da una pulizia del training set in cui vengono eliminate le feature completamente nulle, per evitare qualsiasi problema di apprendimento. Una volta eseguito questo passaggio, il classificatore è pronto per effettuare il training con solo le feature utili ottenendo:

- una misura di accuratezza del training appena effettuato, considerando come inlier tutti i campioni d'ingresso (mentre sappiamo che l'SVM può tenere in considerazione alcuni outliers nel training set con il parametro ν)
- un modello della frontiera selezionata in funzione dei vettori di supporto
- una rappresentazione esplicita dei vettori di supporto utilizzati per descrivere la frontiera

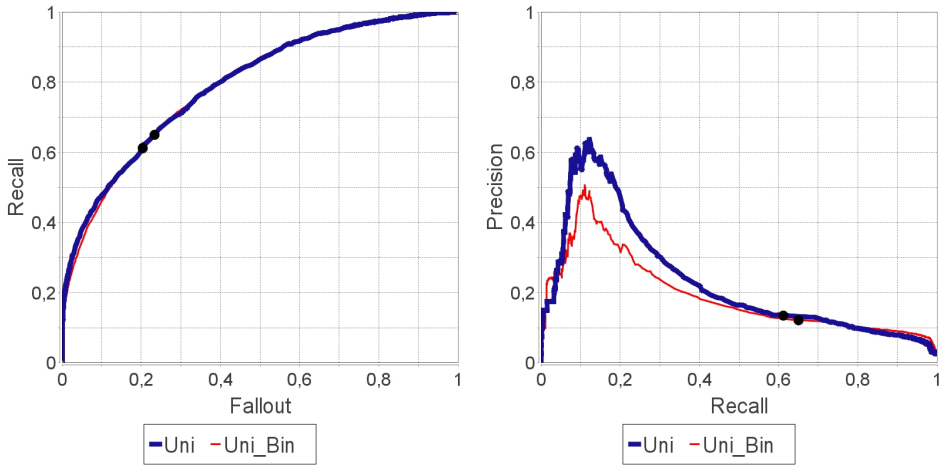
Questo passaggio coclude la fase di addestramento; ad ogni item corrisponde ora un modello che rappresenta la distribuzione dei campioni di training e sarà quindi in grado di decidere se un nuovo tweet appartiene a tale classe o no.

5.2 Analisi feature singole

In questa sezione analizzeremo qualitativamente e quantitativamente i risultati del sistema al variare di molti parametri. Come termini di paragone utilizzeremo le metriche e le curve definite in (§ 3.2). Nella Sezione 4 vedremo le prestazioni dell'algoritmo di base con il quale confrontarci, in §5.4 proveremo a ricavare dei risultati al variare dei dati che abbiamo a disposizione, mentre in §4.2 e §5.3 vedremo qual è il comportamento del sistema modificando il tipo di feature utilizzato. Il lavoro va nella direzione della ricerca di una delle combinazoni migliori di dati e feature per effettuare un confronto finale con l'algoritmo di base.

In questa sezione vedremo quali sono le prestazioni del nostro sistema. In §5.2.1 viene dato un po' di spazio alla ricerca di feature migliorate utilizzando alcuni vincoli, infine in §5.3 le feature vengono aggregate con un politica greedy: partendo dalle feature singole con prestazioni migliori, cerchiamo di aggiungere una ad una le altre per cercare di ottenere risultati sempre migliori. I successivi test riguardano la variazione del volume dei tweet di training (§5.4.1) e del numero di item (§5.4.2). Sucessivamente, nella Sezione 5.2vengono prese in considerazione le feature prese singolarmente e valutate sia in versione binaria che in versione reale (con l'utilizzo del peso IDF).

In questa sezione andremo ad analizzare l'accuratezza (in termini di grafici e metriche standard) del sistema al variare del tipo di feature utilizzato. Per i test successivi sono stati utilizzati tutti i 40 item a disposizione e, per metterci in una situazione realistica, abbiamo addestrato i classificatori su 400 tweet. Nella prima fase (§4.2) abbiamo cercato di capire se e come



(a) ROC (b) Precision/Recall

Figura 5.2: Confronto tra unigrammi in versione IDF e in versione binaria

influisse l'utilizzo di feature binarie rispetto a quelle IDF nei diversi casi; successivamente in §5.2.1 abbiamo provato ad esplorare una via per migliorare le prestazioni delle singole feature. Alla fine di questo processo saremo in grado di decidere quale forma di ogni tipo di feature sia la migliore se utilizzata singolarmente. Nell'ultima fase abbiamo provato a utilizzare un vettore di feature combinato in cui vengono utilizzati tipi di feature diversi per aumentare le prestazioni in termini di potere discriminatorio e precisione (§5.3).

Unigrammi. Il primo test, quello con i soli unigrammi, ci permetterà di verificare quale forma (binaria o IDF) è la migliore da utilizzare.

Dalla Figura 5.2 possiamo notare che la ROC non varia molto nel passaggio da feature IDF a binare, però è interessante vedere come invece la precision massima del caso IDF è migliore di circa il 14% (da 51% a 65%), con recall pressochè invariata. Il raffronto dei due approcci è basato su quello che è il picco massimo della precision; il classificatore SVM però decide sulla base di una soglia preimpostata uguale a 0; tale soglia è indicata esplicitamente nei grafi dal "cerchio nero pieno". In ognuno dei grafici successivi sapremo quindi in quale punto lavorerebbe il classificatore se non fosse analizzato il comportamento al variare della soglia di decisione. In questo caso specifico degli unigrammi infatti, il valore di soglia ottimale (in termini di precision) per il caso migliore (quello IDF) è 1.13, il che sottolinea come sia necessario essere un po' più restrittivi rispetto a quello che è stato l'addestramento normale del classificatore SVM.

Da quest'analisi possiamo decidere che la forma migliore per le feature degli unigrammi è quella IDF; che è la forma che useremo nei prossimi test e che chiameremo semplicemente Uni. Il motivo principale per cui funziona meglio la forma IDF è che, anche se il calcolo della sola presenza di un termine nei tweet (forma binaria) sia efficace per un testo breve, senza una statistica come l'IDF non saremmo in grado di pesare di meno le parole che compaiono frequentemente all'interno dell'insieme degli item. Anche se una parola non è una stopword in generale, alcuni termini come "movie" oppure "DVD" sono sicuramente troppo frequenti in questo settore e possono essere considerate delle stopword relative al contesto, che non aiutano il classificatore ad avere un'alta precisione, a cui di conseguenza si attribuisce un valore in proporzione molto basso.

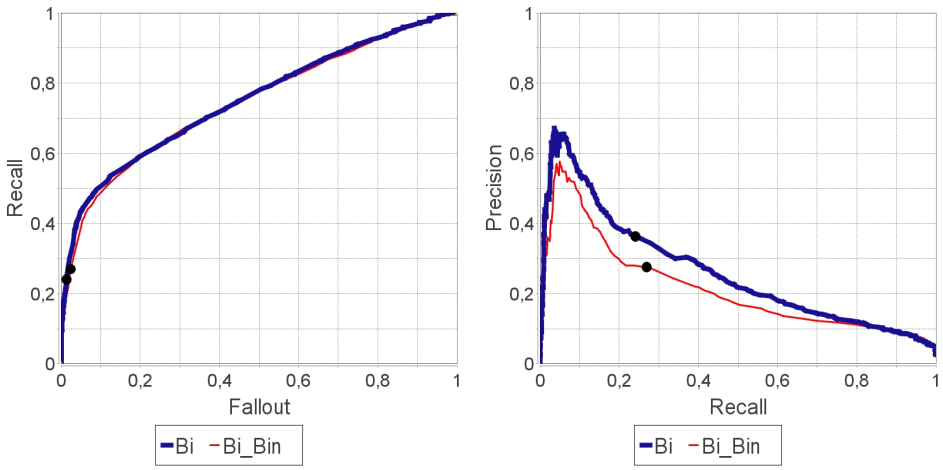
Bigrammi. La stessa analisi viene effettuata per i bigrammi. Anche in questo caso il confronto relativo è lo stesso: la Figura 5.3 ci mostra come la versione IDF per i bigrammi è superiore di circa il 10% rispetto a quella binaria. Sono molto interessanti però alcune osservazioni:

- il valore massimo di precision (67%) è leggermente migliore (+ 2%) rispetto al caso degli unigrammi, in quanto un bigramma molto frequente è più significativo
- la recall nel punto di massima precision è però molto basso (4%) perchè è sempre molto difficile trovare bigrammi significativi e soprattutto in un testo così breve

Nel seguito del lavoro utilizzeremo sempre la forma IDF per i bigrammi, chiamandola Bi, perchè ancora una volta è la versione con le prestazioni migliori in termini di precision.

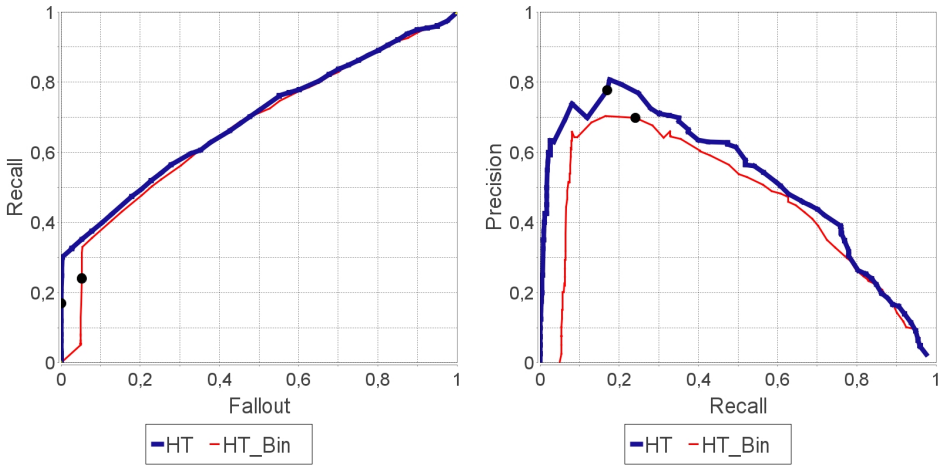
Hashtag. Ancora con lo stesso principio vediamo quale forma è migliore nel caso degli hashtag.

In Figura 5.4 vengono mostrati i risultati. Il caso degli hashtag è particolare: in questo tipo di feature ricadono tutti quei termini che sono una caratteristica di Twitter, perchè sono strettamente correlati a come gli utenti utilizzano il social network, creando e diffondendo anche gli hashtag. É evidente che a fronte di una ROC discreta otteniamo una recall solo del 17% nel punto di massima precision (82%). Questo comportamento è da spiegare con la netta potenza discriminativa degli hashtag e dalla loro relativamente bassa presenza all'interno dei tweet. Questa alta precisione spiega in parte anche i risultati molto alti dell'algoritmo banale: seppur per una porzione



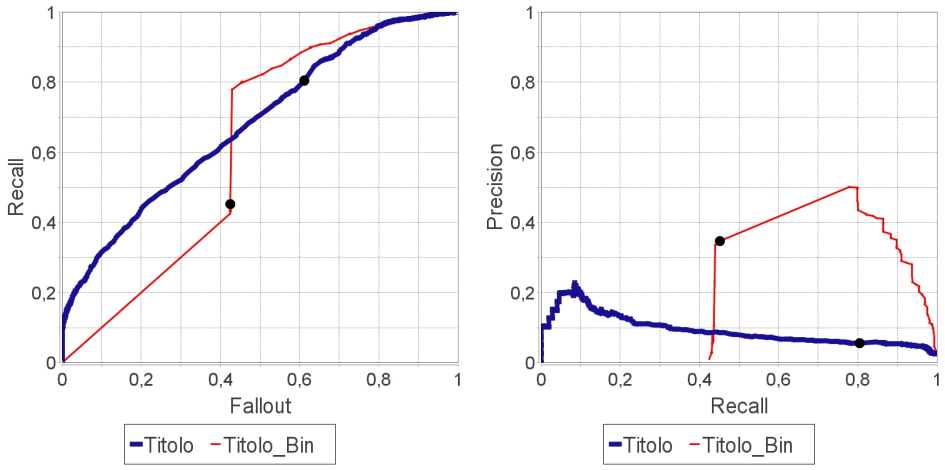
(a) ROC (b) Precision/Recall

Figura 5.3: Confronto tra bigrammi in versione IDF e in versione binaria



(a) ROC (b) Precision/Recall

Figura 5.4: Confronto tra hashtag in versione IDF e in versione binaria



(a) ROC (b) Precision/Recall

Figura 5.5: Confronto della feature ẗitoloïn versione IDF e in versione binaria

discreta di tweet, abbiamo un'informazione molto discriminante. La differenza tra questo approccio e quello dell'algoritmo banale è che in questo caso stiamo apprendendo gli hashtag frequenti e non solo quelli che contengono le parole del titolo. Ad esempio un hastag come *#juliaroberts* non sarà utilizzato dall'algoritmo di base ma sarà appreso (se frequente) dall'algoritmo SVM per tutti i film in cui compare Julia Roberts, con un peso molto maggiore degli unigrammi Julia, Roberts e del bigramma Julia Roberts.

Il risultato di quest'analisi ci porta ad utilizzare ancora una volta la versione IDF anche per gli hashtag, che nel seguito del lavoro chiameremo HT.

Titolo. Nel caso di questa particolare feature possiamo ancora verificare quale sia la forma migliore. C'è però da fare qualche precisazione: in questo tipo di feature non esiste una versione IDF in quanto il valore della feature stessa è una percentuale relativa alla presenza delle parole del titolo nel tweet. Utilizziamo quindi come forma reale (IDF) il valore di tale percentuale, mentre utilizziamo la soglia 0.75 sul valore della feature come discriminante per assegnare un valore binario. La scelta di tale particolare soglia rispetto al valore 1 (che corrisponde al match di tutti i termini del titolo), è basata sui risultati dell'algoritmo di base ottenuti in §4; vogliamo in questo caso SVM consentire al classificatore di apprendere in modo da generalizzare a sufficienza.

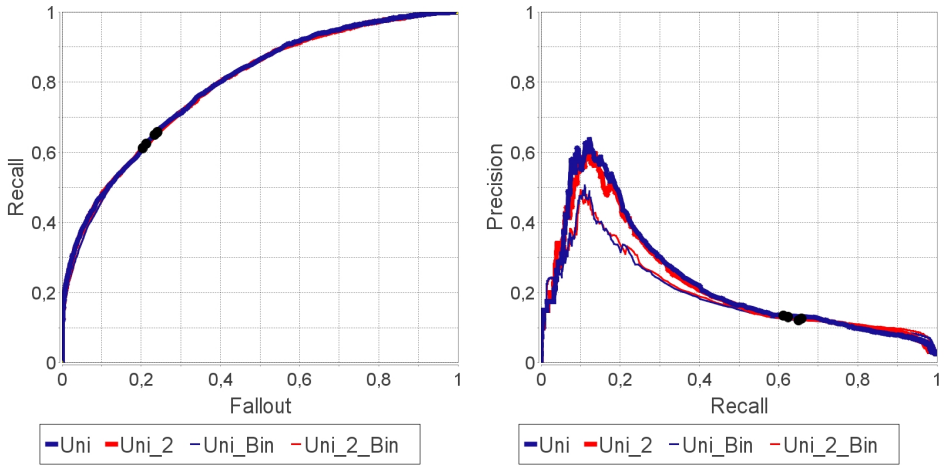
Il comportamento della feature che vediamo in Figura 5.5 è molto particolare: la ROC ci mostra come sia possibile per l'algoritmo binario superare

le prestazioni in termini di potere discriminativo, infatti al variare della soglia il sistema diventa sempre migliore, fino a raggiungere un punto in cui si ha un drastico aumento della recall con un fallout che rimane pressochè invariato. Questo comportamento è dovuto al fatto che la feature titolo funziona molto bene quando la soglia di decisione è un determinante per la classificazione (caso binario). Nel caso IDF invece, il classificatore contiene anche deboli informazioni nelle feature relative ad item (film) con parole in comune, che rovinano le prestazioni. Come previsto riusciamo ad ottenere una buona precision massima del 50% con un'alta recall (80%). Nonostante l'apprendimento, le feature del titolo proposte in questa fase non sono in grado di ottenere risultati paragonabili all'algoritmo di base in quanto il classificatore tende per sua natura a generalizzare, includendo nei risultati molti falsi positivi. In definitiva, la versione binaria di questa feature ricalca in modo migliore l'algoritmo di base, ottenendo migliori prestazioni: possiamo scegliere la versione binaria della feature titolo (che chiameremo semplicemente Titolo) come riferimento per il resto dell'analisi.

5.2.1 Miglioramenti

I risultati della sezione precedente non sono però sufficienti a concludere l'analisi delle feature singole. Prendiamo ora in considerazione un altro parametro: proviamo ad eliminare dal vettore di feature tutti quei termini che compaiono solo 1 o 2 volte. In questa fase proviamo ad alzare il limite di questa soglia da 1 a 2 solo per vedere il comportamento degli unigrammi sottoposti a questo nuovo vincolo; se il risultato sarà migliorativo, proseguiremo l'analisi in questa direzione con soglie ancora diverse e per tutti i tipi di feature.

Nella Figura 5.6 abbiamo chiamato Uni_2 e Uni_2_Bin le versioni in cui eliminiamo i termini a bassa frequenza. Le linee blu del grafico sono le feature normali, le linee rosse rappresentano il tentativo di miglioramento. La ROC è sostanzialmente identica, mentre si può vedere che non solo la precisione non sale in queste versioni, ma peggiora leggermente. Sia nella forma binaria che in quella IDF, non otteniamo risultati apprezzabili in termini di precisione; questo è dovuto principalmente al fatto che in un sistema di feature in cui i termini vegono analizzati dal punto di vista statistico, eliminare quelli con conteggio basso significa eliminare dal classificatore dei termini rari che però, se presenti in un solo item, sono elementi di forte discriminazione. Rimuovento i termini con conteggio minore o uguale a 2 non peggiora le prestazioni di molto ma non otteniamo nessun miglioramento, neppure nel caso binario (in cui il peggioramento è quasi impercettibile). Dopo que-



(a) ROC (b) Precision/Recall

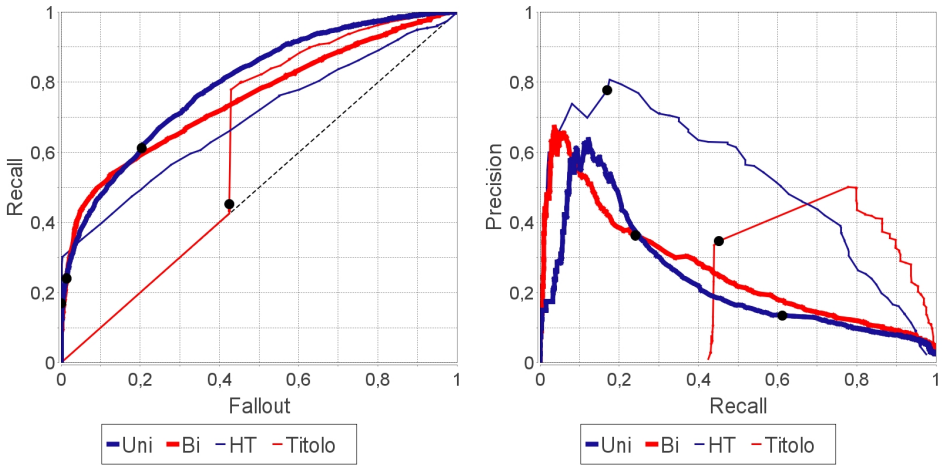
Figura 5.6: Confronto degli unigrammi rispetto all'eliminazione dei termini a bassa frequenza (caso IDF e binario)

st'analisi siamo in grado di scegliere ognuna delle feature nella sua forma corretta e di scartare il tipo di miglioramenti provato in questa sezione.

5.2.2 Discussione

In base a qunto visto finora, possiamo dire in quali casi è meglio utilizzare una feature binaria rispetto a quella IDF e se è utile eliminare i conteggi bassi. Il prossimo passo è quello di verificare quale delle feature funzioni in modo migliore se presa singolarmente.

Il confronto in Figura 5.7 evidenzia una comportamento nettamente migliore da parte delle feature che utilizzano gli hashtag che raggiungono una precision dell'82% con una recall del 18%. La ROC ci mostra come in teoria il comportamento di unigrmmi e bigrammi possa apparire migliore degli hashtag; c'è da considerare che, vista la recall bassa, il punto di lavoro che stiamo considerando (quello a massima precision) si trova all'inizio della curva, quasi in corrispondenza del punto nero che evidenzia la soglia 0. In questa porzione di ROC si vede come gli hashtag ottengano un fallout quasi nullo, mentre unigrammi e bigrammi inizino ad ottenere una crescita del fallout. A questo livello di analisi possiamo quindi dire che le feature migliori sono quelle che utilizzano gli hashtag.



(a) ROC

(b) Precision/Recall

Figura 5.7: Confronto tra le feature utilizzate singolarmente

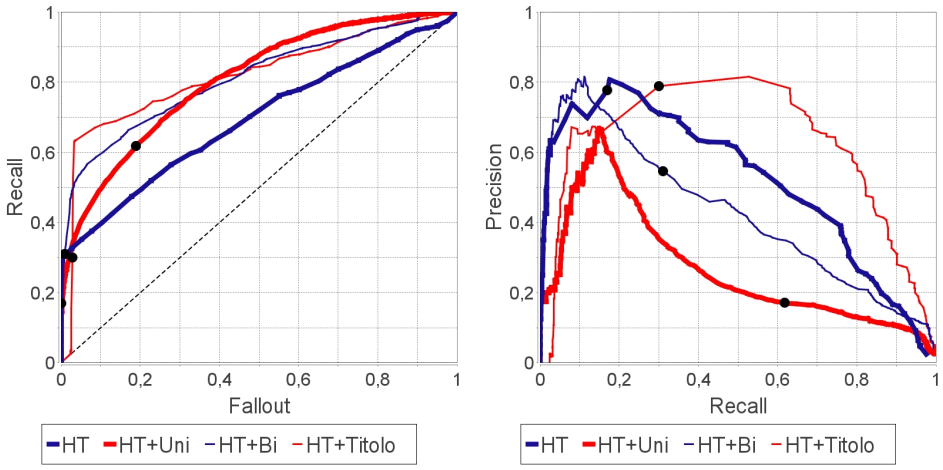
5.3 Analisi feature aggregate

Come abbiamo appena visto, le feature che raggiungono la precisione più elevata con una curva ROC accettabile sono gli Hashtag. Da questo punto in poi le feature vengono combinate per cercarne una versione migliore. Nonostante sia possibile (anche se improbabile) che le feature migliori non contengano gli hashtag, per non esplorare tutto l'enorme spazio di combinazioni, utilizziamo una approccio greedy. Lavoriamo a fasi successive cercando di arricchire in ogni fase la feature migliore per la fase precedente. In tutto questo processo vengono utilizzate le feature nella loro forma migliore ricavata in §4.2.

Feature di 2 tipi. In questa fase confronteremo la feature singola migliore (gli Hashtag) con le sue 3 possibili estensioni:

- Hashtag + Unigrammi
- Hashtag + Bigrammi
- Hashtag + Titolo

I risultati di questa fase sono mostrati in Figura 5.8. Nessuna delle estensioni delle feature degli hashtag migliora il valore della massima precision (82%) già precedentemente ottenuto dagli hashtag utilizzati singolarmente. In questa situazione andiamo a privilegiare la combinazione di feature che ci permette di lavorare ad una precision vicina a quella massima, con



(a) ROC (b) Precision/Recall

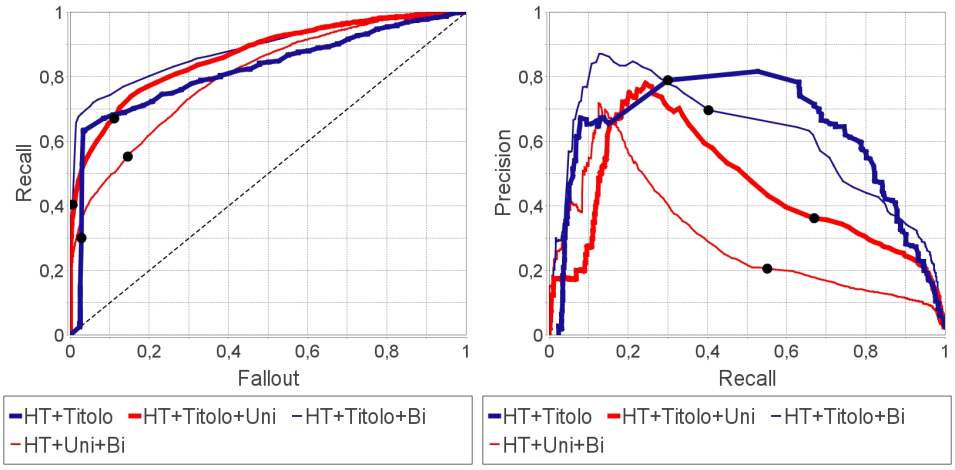
Figura 5.8: Confronto tra gli hashtag e le sua possibili estensioni

la recall migliore, ovvero HT+Titolo. La ROC ci conferma come il potere discriminativo degli hashtag sia migliorato nel modo migliore dalla versione HT+Titolo; questo è dovuto al fatto che con questo tipo di feature riusciamo ad ottenere un'altra precision massima e a includere i risultati di alta precision delle feature Titolo. Nonostante l'utilizzo di questa feature utilizzi quasi gli stessi dati a disposizione dell'algoritmo base, non riusciamo in questa versione SVM a raggiungere i suoi risultati; anche se la recall migliore delle feature HT+Titolo è circa il 60%, in questo punto otteniamo una precision vicina all'80%, ancora il 16% in meno dell'algoritmo base.

Feature di 3 tipi. Proviamo ad aggiungere ulteriori informazioni per aiutare il sistema di classificatori SVM a ottenere risultati migliori. Utilizziamo le feature migliori della prima fase (HT+Titolo) e valutiamo le 2 possibili estensioni più una terza versione in cui non utilizziamo il titolo per verificare la variazione di prestazioni:

- Hashtag + Titolo + Unigrammi
- Hashtag + Titolo + Bigrammi
- Hashtag + Unigrammi + Bigrammi

Come si vede in Figura 5.9, in questo caso abbiamo un aumento della precision massima nel caso HT+Titolo+Bi che arriva fino all'88%. Il prezzo da pagare per una precision migliore è nella recall che scende fino al 12%;



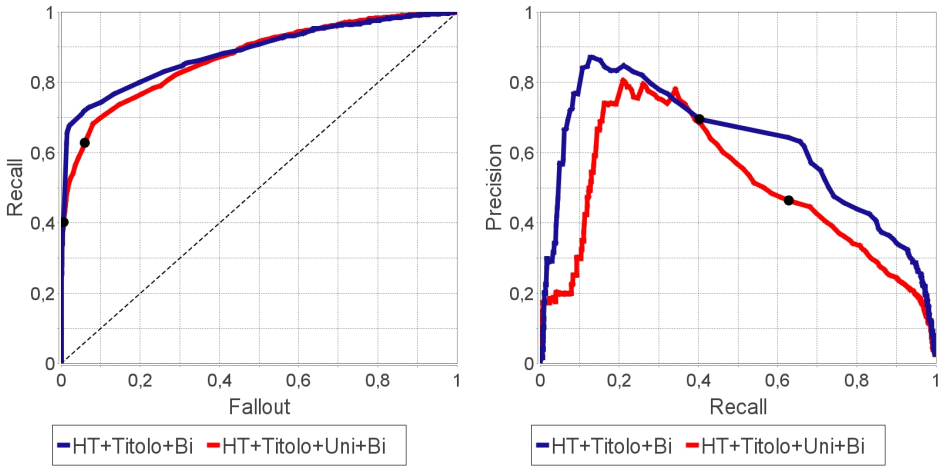
(a) ROC (b) Precision/Recall

Figura 5.9: Confronto tra HT+Titolo e alcune possibili varianti

è interesante però notare come con l'approccio misto HT+Title+Bi siano evidenti i punti di forza di tutte le componenti: abbiamo raggiunto un miglioramento della precisione massima utilizzando gli hashtag e bigrammi, ma siamo riusciti a migliorare l'andamento della precision in funzione della recall, come nel caso del Titolo. Se da un lato, nella curva Precision/Recall, vediamo che il grafico relativo a HT+Titolo+Bi scende decisamente molto più piano degli altri approcci con 3 tipi di feature, dall'altro abbiamo la conferma della curva ROC che ci mostra come il potere discriminativo di questa combinazione sia il migliore. Si può notare inoltre come l'aggiunta degli unigrammi sia peggiorativa; anche nel caso dei confronti con 2 tipi di feature questo è avvenuto, ma abbiamo dovuto aspettare anche questi test per averne la conferma. Il motivo di questa diminuzione delle prestazioni è probabilmente che gli unigrammi sono poco distintivi per caratterizzare un documento così corto come un tweet; inoltre la capacità di utilizzare i termini singoli più importanti è già presente nelle versioni Titolo e HT. In questo senso è probabile che l'aggiunta degli unigrammi aggiunga principalmente del rumore.

Feature di 3 o 4 tipi. Questa è l'ultima fase dell'analisi delle feature aggregate, in cui confrontiamo le feature migliori note finora (HT+Titolo+Bi) con la versione che del sistema che utilizza tutti i tipi di feature.

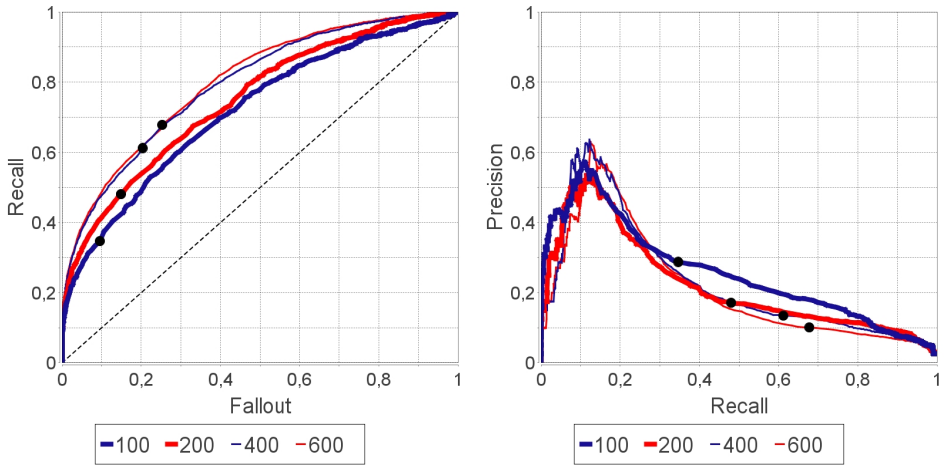
Anche in questo ultimo caso, come visto nelle sezioni precedenti, l'aggiunta degli unigrammi peggiora le prestazioni. In Figura 5.10 si può vedere che sia nella ROC che nella curva Precision/Recall la differenza è evidente.



(a) ROC

(b) Precision/Recall

Figura 5.10: Confronto finale



(a) ROC (b) Precision/Recall

Figura 5.11: Analisi di sensitività al variare del numeri di tweet disponibili per il training

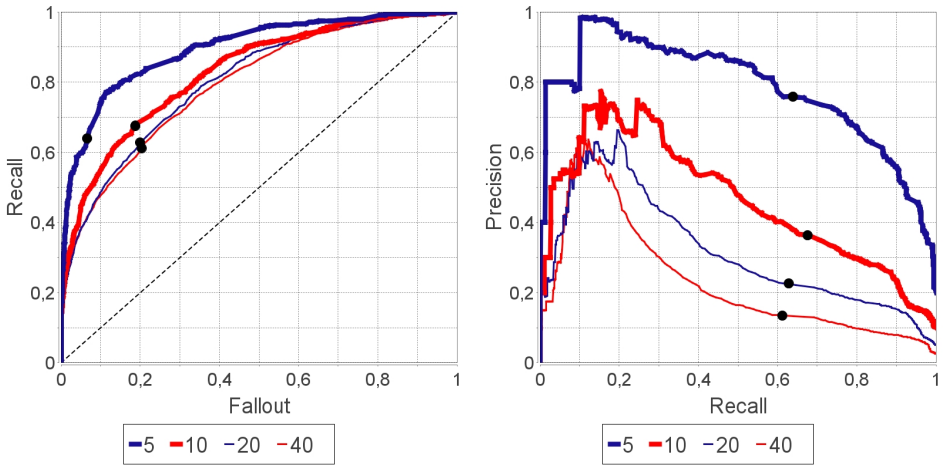
5.4 Analisi di sensitività

L'analisi di sensitività si occupa di capire come e quanto variano le prestazioni del sistema al variare delle informazioni a disposizione. Questo passo ci permette di capire come variano i parametri ambientali nel contesto di utilizzo sul sistema.

5.4.1 Volume dei tweet di training

Il primo parametro che andiamo ad analizzare è il numero di tweet che viene utilizzato nella fase di training. Questo parametro può essere decisivo in quanto è una misura della quantità di informazioni che il classificatore ha a disposizione per l'apprendimento e che influirà sulle prestazioni. É importante valutare tale aspetto per vedere se e quando il classificatore va in over(under)fitting e decidere quale sia la soglia minima o massima di tweet necessari per un corretto addestramento. Sarà anche possibile valutare se il numero di tweet a disposizione è sufficiente per un'analisi che potrebbe anche essere limitata temporalmente. In questo contesto è importante che i tweet siano distribuiti uniformemente e quindi ci aspettiamo che al crescere del volume del training set si possa ottenere un classificatore sempre migliore. Il test è stato esguito su tutti i 40 items e per le sole feature degli unigrammi in versione IDF.

I 2 grafici in Figura 5.11 mostrano i risultati di quest'analisi: entrambe le curve mostrano come sia possibile un miglioramento all'aumentare del



(a) ROC (b) Precision/Recall

Figura 5.12: Analisi di sensitività al variare del numeri di item nel sistema

numero di tweet disponibili per effettuare il training. Questo situazione riflette il comportamento normale di un classificatore: all'aumentare dei dati disponibili il classificatore otterrà prestazioni migliori, fino ad arrivare ad una condizione di overfitting. La considerazione particolare di questo caso specifico è però che c'è un comportamento di saturazione delle prestazioni tra l'utilizzo di 400 e 600 tweet; questo potrebbe essere il limite oltre il quale i nuovi tweet aggiungono solo rumore oppure non aggiungono informazioni utili.

5.4.2 Numero di items

Il secondo test di sensitività viene effettuato al variare del numero di item a disposizione. Anche in questo casi siamo interessati a vedere l'andamento delle curve ROC e PR quando il numero di item nel sistema è più o meno grande. Sebbene il sistema sia ideato per un numero di item arbitrariamente alto, questo test è stato effettuato fino ad un massimo di 40 item (quelli a nostra disposizione) per vedere l'andamento e capire se esistono dei comportamenti di saturazione rispetto ad un limite. In questo caso sono stati utilizzati 400 tweet per il training e sempre per le sole feature degli unigrammi in versione IDF.

I risultati del test (in Figura 5.12) mostrano la difficoltà del sistema a classificare correttamente i tweet quando siamo in presenza di un numero di item elevato. Nella ROC si può notare lo stesso comportamento di saturazione che abbiamo esaminato nel precedente test di sensitività: in questo

caso però le prestazioni scendono fino ad un limite oltre il quale l'aggiunta di nuovi item potrebbe smettere di influire negativamente sul sistema. La curva Precision/Recall evidenzia un comportamento simile ma in questo caso l'andamento della precision al variare della soglia decresce continuamente; in questo caso però è opportuno guardare il valore massimo della precision, che conferma il comportamento di saturazione.

5.5 Discussione finale

L'analisi delle possibili configurazioni in cui far lavorare il sistema di classificazione basato su SVM ci ha mostrato come non sia possibile per ora raggiungere o migliorare le prestazioni dell'algoritmo di base. Nonostante l'utilizzo di feature combinate, siamo riusciti a raggiungere un risultato massimo di pecision dell'88%, contro un massimo di 96% per l'algoritmo di base. Per ottenere risultati paragonabili anche in termini di recall, dobbiamo eliminare dalle feature i bigrammi e scendere fino ad una precision massima dell'80%. Concludendo, non è stato possibile utilizzare feature linguistiche in un sistema basato su una batteria di classificatori one-class SVM per ottenere un miglioramento delle prestazioni dell'algoritmo di base.

Capitolo 6

La soluzione avanzata

Alla luce dei risultati del Capitolo 5 di questo lavoro non siamo stati in grado di migliorare le prestazioni dell'algoritmo di base utilizzando i classificatori SVM proposti nel Capitolo 5. Questo nuovo approccio (Figura 6.1 solo livello 1 e 2) si propone di utilizzare congiuntamente i risultati ottenuti dai due algoritmi per ottenere delle prestazioni ancora superiori. Seppur con un'elevata precisione, l'algoritmo di base raggiunge una recall non elevatissima; sfruttando il nostro approccio nei casi in cui l'algoritmo di base fallisce, è possibile che vengano raggiunte prestazioni superiori.

Il nuovo sistema segue l'architettura del sistema della fase 1, ma in questo caso al posto di ogni classificatore normale vi sarà un sistema di classificazione combinato che restituirà diversi valori di decisione per ogni tweet di ingresso. Avremo quindi più informazioni per decidere la classe di appartenenza di ogni singolo tweet.

Nelle prossime sezioni analizzeremo l'approccio a doppio stage §6.1 e un metodo basato su alcune politiche più avanzate per la scelta della classe corrispondente §6.2.

6.1 Algoritmo a doppio livello

Questo è l'approccio più basilare per utilizzare i due tipi di classificazione nello stesso sistema. Ogni classificatore è sostituito da un metodo di classificazione a doppio livello:

- livello 1: ogni tweet d'ingresso viene analizzato con l'algoritmo di base rispetto a tutti gli item e vengono classificati positivamente già in questo livello tutti i tweet che superano la soglia prescelta (nel nostro caso 0.75)

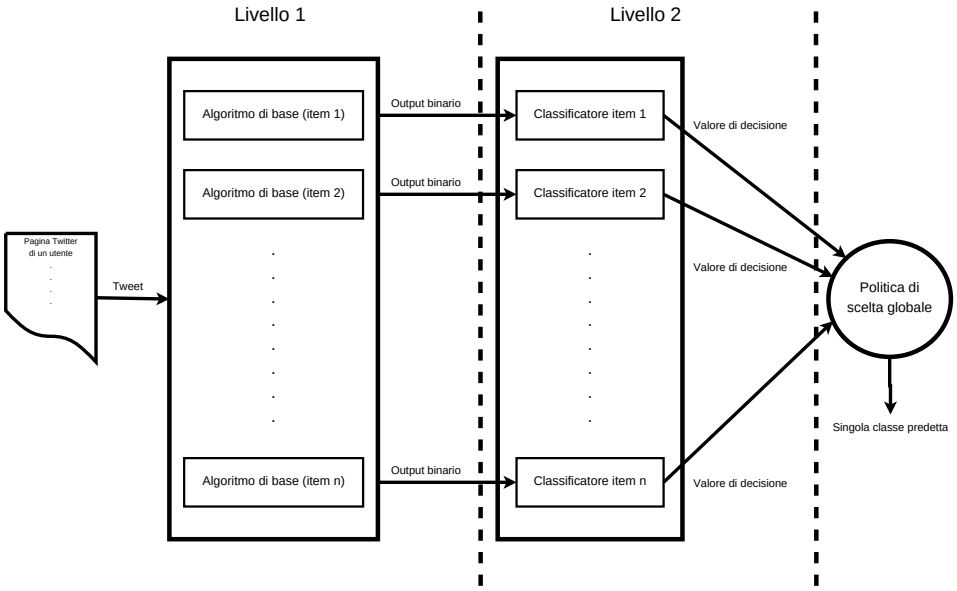


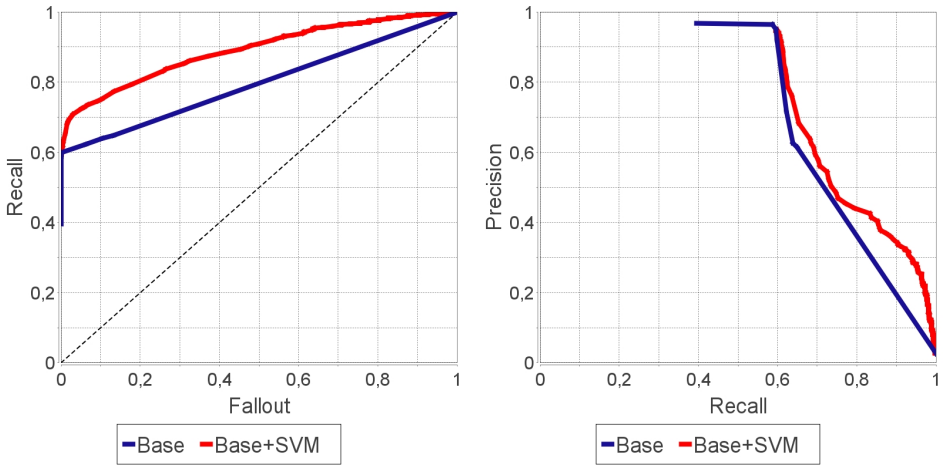
Figura 6.1: Sistema a 2 livelli (con politica di scelta globale)

- livello 2: i restanti tweet vengono analizzati con ognuno dei classificatori del sistema per estrarne un valore di decisione

In questo caso otterremo per ogni coppia tweet/item non più un singolo valore ma due, ognuno relativo ad un livello. Tale procedura inserisce una precedenza nell'utilizzo dei due algoritmi, preferendo quello di base a fronte della sua alta precision; i valori di decisione vengono utilizzati nei casi in cui il primo livello (l'algoritmo di base) non sa rispondere. Questa soluzione permette ancora a più classificatori di rispondere positivamente quindi, nonostante le possibili migliori prestazioni, deve essere adattata al caso reale in cui ado ogni tweet corrisponde al massimo una classe (item); questo sarà esplorate nella Sezione 6.2. Per rendere confrontabile questa nuova tecnica con le predenti vengono infine calcolate le curve ROC e PR variando sempre la soglia del valore di decisione dei classificatori.

Risultati. Lo scopo di questo test è quello di ottenere dei risultati per l'algoritmo combinato da poter confrontare con gli approcci precedenti, quindi ammettendo le risposte positive multiple per un singolo tweet. Il valore di soglia utilizzato come parametro lineare nella ROC e nella PR ci permette di:

- ottenere i risultati relativi al solo livello 1, impostando la soglia massima



(a) ROC

(b) Precision/Recall

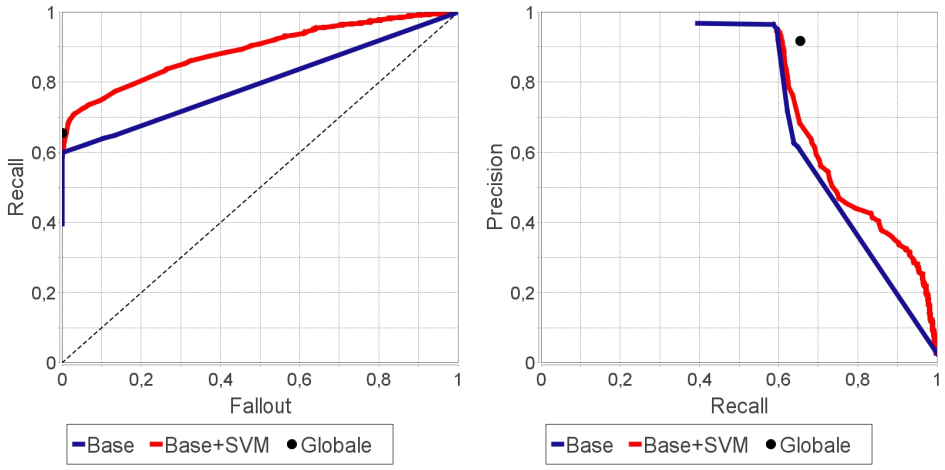
Figura 6.2: Sistema combinato Base+SVM (a due livelli)

- variare la soglia per modificare il comportamento e l'accuratezza del sistema di classificazione che compone il livello 2 al fine di migliorare le prestazioni globali
- valutare il punto di lavoro migliore del livello 2 in funzione del livello 1

In Figura 6.2 sono mostrati i risultati dell'esperimento. Unendo i due approcci in un algoritmo a doppio livello non siamo in grado di migliorare le prestazioni di precision, in quanto la recall dei bigrammi è troppo bassa per dare un apporto significativo. Questo esperimento però ci permette di migliorare l'andamento della precision in funzione della recall dell'algoritmo base: mentre modificare la soglia dell'algoritmo base (la variabile lineare della serie blu) peggiora drasticamente le prestazioni, la modifica della soglia di decisione del classificatore (la variabile lineare della serie rossa) ci permette di valutare come sia possibile aumentare la recall del sistema, senza peggiorare drasticamente in termini di precision come nell'algoritmo di base. Questo comportamento si riflette anche in un netto miglioramento della curva ROC.

6.2 Politica di scelta globale

L'architettura del sistema di classificazione è ancora quella in Figura 6.1, però in questo caso cerchiamo di interpretare il doppio valore in uscita da ogni classificatore come valore locale e andiamo a scegliere una singola classe



(a) ROC

(b) Precision/Recall

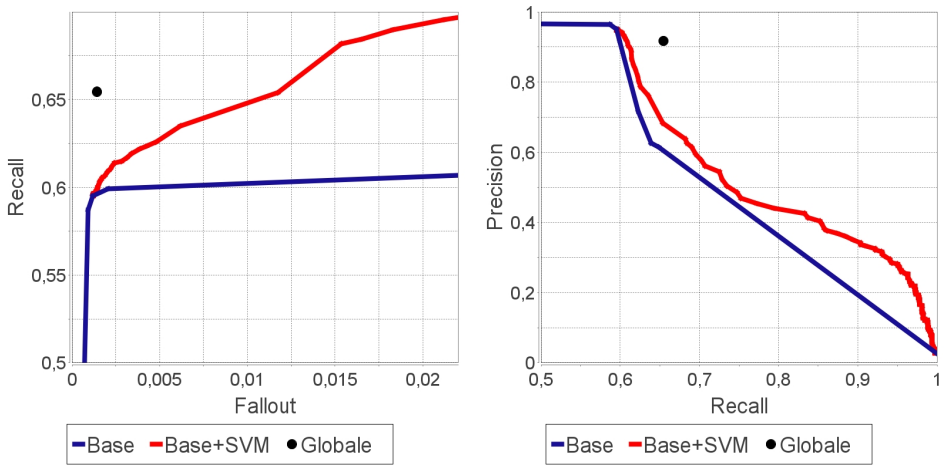
Figura 6.3: Punto di lavoro del sistema globale

per ogni tweet di ingresso valutando ogni output di ogni classificatore a cui viene sottoposto come input. Questo scenario si adatta bene al caso in cui vogliamo realemente scegliere al massimo una classe per ogni tweet e eventualmente integrarlo con un sistema successivo come può essere un sistema si sentiment analysis oppure un sistema di raccomandazione. In questo ultimo caso non abbiamo una soglia di decisione da fare variare per misurare le prestazioni, quindi calcoleremo i valori delle metriche Precision, Recall e Fallout in modo globale e li confronteremo con le massime prestazioni degli approcci precedenti.

Risultati. I risultati finali di questo tipo di approcci sono mostrati in Figura 6.3, in cui possiamo vedere quale punto di lavoro riusciamo a raggiungere utilizzando una politica di scelta della classe corrispondente ad un tweet di tipo globale. Entrambi i grafici ci spiegano come sia possibile aumentare sensibilmente la recall del sistema globale senza avere un notevole peggioramento della precision (che rimane sopra il 90%).

Per cogliere i dettagli e le caratteristiche del sistema globale osserviamo meglio il punto di ottimo nella Figura 6.4. Guardando attentamente la posizione del punto nero possiamo quindi confermare che:

- siamo riusciti ad aumentare la recall, mantenendo un fallout molto basso (Figura 6.4(a))
- la recall aumenta con una diminuzione della precision molto piccola (Figura 6.4(b))



(a) ROC (Versione Zoomata)

(b) Precision/Recall (Versione Zoomata)

Figura 6.4: Punto di lavoro del sistema globale (Zoom sul punto di lavoro)

	Precision (%)	Recall (%)	Fallout (%)
Advanced matching	96	58	circa 0
HT+Titolo+Bi (SVM)	88	12	circa 0
SVM a due livelli	96	58	circa 0
SVM globale	92	65	circa 0

Tabella 6.1: Confronto tra i principali metodi utilizzati

La precision non è l'unico parametro da tenere in considerazione per valutare un sistema così articolato, talvolta può essere necessario ottenere miglioramenti in termini di recall anche a discapito di una piccola variazione di precision. Questo sistema globale ci ha permesso infatti di migliorare le prestazioni dell'algoritmo banale arrivando ad una recall del 65% con una precision del 92%.

6.3 Discussione finale

Nella Tabella 6.1 vengono elencati i principali metodi studiati in questa tesi, con i corrispondenti valori delle metriche principali. L'algoritmo di base ci ha sorpreso raggiungendo un'altissima precision, mentre l'algoritmo SVM ha raggiunto una buona precision (anche se non sufficiente a migliorare l'algoritmo di base) ma a discapito di una forte diminuzione della recall. Utilizzando il sistema a 2 livelli siamo riusciti ad analizzare il comportamento dei valori delle metriche in gioco, anche se le migliori prestazioni sono quelle

del solo livello 1 (l'algoritmo base). Introducendo infine una strategia di scelta basata sull'utilizzo dei valori del livello 2, siamo riusciti ad ottenere una precision molto vicina alla migliore, migliorandone le recall in modo sensibile.

Capitolo 7

Conclusioni e sviluppi futuri

L'obiettivo di questa tesi era quello di costruire un sistema in grado di classificare tweet in base al loro item di appartenenza. Abbiamo costruito un primo sistema di base che utilizzava metodi semplici di keyword extraction, ottenendo risultati sorprendenti (96% di precision e 58% di recall).

Partendo da questo approccio base, abbiamo definito un sistema basato su una batteria di classificatori one-class SVM. Il sistema basato solo su classificatori SVM non è stato in grado di raggiungere le prestazioni della tecnica base Quindi, per cercare di migliorare le misure di prestazioni abbiamo utilizzato un approccio a 2 livelli che integrasse i sistemi precedentemente studiati, scoprendo che utilizzando una strategia globale di scelta della classe corretta sul sottosistema SVM, otteniamo un discreto aumento della recall (65%) con una minima diminuzione della precision (92%).

Sviluppi futuri. Il progetto proposto in questa tesi è composto da alcuni algoritmi che sono stati sviluppati cercando di analizzare il maggior numero di variabili possibile. Lo spazio delle combinazioni di tali variabili è molto elevato, quindi per questo studio iniziale non le abbiamo esaminate tutte, ma è possibile che la modifica di alcune di queste possano produrre dei risultati migliori. Vista la quantità di dimensioni esplorabili con questo tipo di approccio, elenchiamo in ordine logico le varianti ancora esplorabili:

- modificare, migliorare e rendere più specifica rispetto al contesto la lista delle stopword
- definire e valutare un metodo più avanzato di decisione della soglia anche per l'algoritmo base, eventualmente anche dipendente dall'item in questione

- utilizzare un altro metodo per il peso relativo dei termini in un documento rispetto al peso IDF (§ 5.1.2) per provare a migliorare le prestazioni nel caso specifico di Twitter

- modificare, semplificandolo, lo spazio delle feature con dei metodi matematici, come SVD

- analizzare a fondo lo spazio dei parametri che modificano il comportamento del classificatore (ν e γ) per verificare la presenza di altre regioni ottime o sub-ottime all'interno delle quali cercare la miglior coppia di parametri

- usare una diversa funzione kernel per il classificatore SVM

- estendere l'analisi nel contesto degli item utili per un futuro sistema di raccomandazione cercando di verificare il comportamento all'aumentare del numero di item a dosposizione

- esplorare in modo approfondito e completo lo spazio delle possibili combinazioni di feature

- trovare un nuovo metodo per aggregare e utilizzare simultaneamente i dati relativi all'algoritmo di base e quelli dell'algoritmo svm, ad esempio con un valore aggregato in basi ad alcuni pesi piuttosto che un doppo stage

- formalizzare e verificare le prestazioni di una nuova politica di scelta di una classe singola per ogni tweet

Oltre a questa esplorazione alla ricerca di ulteriori valori per le variabili in gioco che possano ottimizzare l'intero processo, è interessante vedere quali possono essere le nuove idee per integrare, aggiungere o modificare alcune parti del sistema.

Come primo approccio si potrebbe approfittare della struttura del tweet e dei metadati contenuti nelle risposte HTTP in modo da utilizzare più informazioni rispetto alla semplice estrazione degli hashtag, visto che uno dei vincoli è quello dei 140 caratteri e la necessità di informazioni aggiuntive è cruciale nell'ambiente del microblogging. Il punto comune di partenza potrebbe essere in ogni caso un hashtag, da cui partire per esplorare le connessioni semantiche che questo ha con altri hashtag o metadati.

Un campo molto promettente di sviluppo è quello della costruzione di feature nuove, magari più adatte all'utilizzo nel campo del microblog e più

ïnformativeösotto questo punto di vista. Il problema da fronteggiare in questo settore è quello del rumore che i tweet possono contenere, data la loro natura ïmmediata; ïliberaë sociale. Un'idea per far fronte a questo problema potrebbe essere quella di utilizzare metriche di distanza tra stringhe di testo, come ad esempio la distanza di Levenshtein, che siano in grado di ovviare al problema dello slang piuttosto che a quello delle abbreviazioni o agli errori nella composizione dei messaggi. É chiaro che questa è solo un'alternativa ai classici metodi basati sulla linguistica per correggere gli errori nel testo, che potrebbero essere inclusi e valutati in ogni caso.

In alcuni sistemi più complessi potrebbe essere utile approfittare della natura dei tweet per analizzarli anche in modo temporale: è molto frequente nei social network in generale, che vi sia un comportamento di massaïella pubblicazione dei commenti a qualche evento in particolare, come potrebbe essere l'uscita di un nuovo film. Inoltre è anche possibile localizzare geograficamente alcuni tweet, aprendo un'ulteriore dimensione spaziale oltre a quella temporale. Per questi scopi potrebbe essere anche necessario introdurre nuove metriche di valutazione delle prestazioni che tengano in considerazione nuove dimensioni, e forse sarà ache necessario utilizzare nuovi metodi per rappresentare i risultati-

Come già spiegato nell'introduzione (Capitolo 1), questo lavoro è stato un'analisi propositiva di una particolare fase di un processo molto complesso e articolato. Il passo più importante sarebbe quello di inserire il sitema sviluppato in un progetto più ampio che prenda in considerazione l'item detection solo come una delle fasi di elaborazione delle informazioni. Il flusso di lavoro che parte dal singolo tweet e arriva fino alla produzione di un particolare rating per un particolare item include necessariamente una fase di item detection, ma questa deve essere integrata con un sistema in grado di arrivare al risultato finale: in questo caso un rating. Lo sviluppo di un sistema di sentiment analysis è sicuramente il passo successivo al lavoro proposto in questa tesi; dopo la nostra analisi, potranno essere raccolti i tweet di un particolare utente in cui viene commentato un certo item, e cercare di estrarre da essi un'opinione. In ultima analisi, questo processo fornisce abbastanza informazioni per poter ricavare, attraverso metodi di integrazione pensati appositamente per un sistema di raccomandazione, un singolo valore di rating per la coppia utente/item, che andrebbe a migliorare le prestazioni del sistema di raccomandazione. In questo contesto, il contributo sarebbe molto positivo, in quanto in parte vengono risolti alcuni dei problemi più gravi e complessi relativi ad un sistema di raccomandazione, come ad esempio quello del nuovo utente.

Bibliografia

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing user modeling on twitter for personalized news recommendations. In *Proceedings of the 19th international conference on User modeling, adaption, and personalization*, UMAP'11, pages 1–12, Berlin, Heidelberg, 2011. Springer-Verlag.

[2] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Semantic enrichment of twitter posts for user profile construction on the social web. In *Proceedings of the 8th extended semantic web conference on The semanic web: research and applications - Volume Part II*, ESWC'11, pages 375–389, Berlin, Heidelberg, 2011. Springer-Verlag.

[3] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[4] David Andrzejewski and Xiaojin Zhu. Latent dirichlet allocation with topic-in-set knowledge. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn '09, pages 43–48, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[5] David Andrzejewski, Xiaojin Zhu, and Mark Craven. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 25–32, New York, NY, USA, 2009. ACM.

[6] Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pa-

ges 36–44, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[7] Adam Bermingham and Alan F. Smeaton. Classifying sentiment in microblogs: is brevity an advantage? In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1833–1836, New York, NY, USA, 2010. ACM.

[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[9] Danah Boyd, Scott Golder, and Gilad Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, HICSS '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[10] Thorsten Brants. TnT - a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, 2000.

[11] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, ANLC '92, pages 152–155, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[12] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. Emerging topic detection on twitter based on temporal and social terms evaluation. In *Proceedings of the Tenth International Workshop on Multimedia Data Mining*, MDMKDD '10, pages 4:1–4:10, New York, NY, USA, 2010. ACM.

[13] S. Chan, B. He, and I. Ounis. An in-depth survey on the automatic detection and correction of spelling mistakes. In *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, 2005.

[14] E. Charniak, C. Hendrickson, N. Jacobson, and M. Perkowitz. Equations for part-of-speech tagging. In *In Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789, 1993.

[15] Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1185–1194, New York, NY, USA, 2010. ACM.

[16] Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 355–362, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[17] Yingxiao Zhang Chunliang Lu, Wai Lam. Twitter user modeling and tweets recommendation based on wikipedia concept graph. *Association for the Advancement of Artificial Intelligence*, 2009.

[18] Paul Cook and Suzanne Stevenson. An unsupervised model for text message normalization. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 71–78, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[20] Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. In *Proceedings of the third conference on Applied natural language processing*, ANLC '92, pages 133–140, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[21] Cassaundra Doerhmann. Named entity extraction from the colloquial setting of twitter.

[22] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 281–285, New York, NY, USA, 1988. ACM.

[23] Asli Çelikyilmaz, Dilek Hakkani-Tür, and Junlan Feng. Probabilistic model-based sentiment analysis of twitter messages. In Dilek Hakkani-Tür and Mari Ostendorf, editors, *SLT*, pages 79–84. IEEE, 2010.

[24] Andrea Esuli and Fabrizio Sebastiani. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 617–624, New York, NY, USA, 2005. ACM.

[25] Michael Gamon and Anthony Aue. Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, FeatureEng '05, pages 57–64, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[26] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[27] A Go, L Huang, and R Bhayani. Twitter sentiment analysis. *Entropy*, 2009(June):17, 2009.

[28] Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. Integrating topics and syntax. In *In Advances in Neural Information Processing Systems 17*, pages 537–544. MIT Press, 2005.

[29] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 368–378, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[30] S.M.S. Hasan and D.A. Adjeroh. Proximity-based sentiment analysis. In *Applications of Digital Information and Web Technologies (ICADIWT), 2011 Fourth International Conference on the*, pages 106 –111, aug. 2011.

[31] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, EACL '97, pages 174–181, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.

[32] Vasileios Hatzivassiloglou and Janyce M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings*

of the 18th conference on Computational linguistics - Volume 1, COLING '00, pages 299–305, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[33] Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Nat. Lang. Eng.*, 11(1):87–111, March 2005.

[34] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Mach. Learn.*, 42(1-2):177–196, January 2001.

[35] Courtenay Honeycutt and Susan C. Herring. Beyond microblogging: Conversation and collaboration via twitter. In *HICSS*, pages 1–10. IEEE Computer Society, 2009.

[36] Wei Jin, Hung Hay Ho, and Rohini K. Srihari. Opinionminer: a novel machine learning system for web opinion mining and extraction. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 1195–1204, New York, NY, USA, 2009. ACM.

[37] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1331–1336. AAAI Press, 2006.

[38] Hiroshi Kanayama and Tetsuya Nasukawa. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 355–363, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[39] Shiva Prasad Kasiviswanathan, Prem Melville, Arindam Banerjee, and Vikas Sindhwani. Emerging topic detection using dictionary learning. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 745–754, New York, NY, USA, 2011. ACM.

[40] M. Kaufmann. Syntactic Normalization of Twitter Messages. *studies*, 2, 2010.

[41] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25:259–284, 1998.

[42] VI Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, 1966.

[43] Hang Li and Kenji Yamanishi. Topic analysis using a finite mixture model. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*, EMNLP '00, pages 35–44, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[44] Si Li, Hao Zhang, Weiran Xu, Guang Chen, and Jun Guo. Exploiting combined multi-level model for document sentiment analysis. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 4141 –4144, aug. 2010.

[45] Juan Antonio Lossio Ventura, Hakim Hacid, Arnaud Ansiaux, and Maria Laura Maag. Conversations reconstruction in the social web. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 573–574, New York, NY, USA, 2012. ACM.

[46] Zhunchen Luo, Miles Osborne, and Ting Wang. Opinion retrieval in twitter. In John G. Breslin, Nicole B. Ellison, James G. Shanahan, and Zeynep Tufekci, editors, *ICWSM*. The AAAI Press, 2012.

[47] Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. Simple semantics in topic detection and tracking. *Inf. Retr.*, 7(3-4):347–368, September 2004.

[48] Hassan H. Malik and Vikas S. Bhardwaj. Automatic training data cleaning for text classification. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining Workshops*, ICDMW '11, pages 442–449, Washington, DC, USA, 2011. IEEE Computer Society.

[49] Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceedings of the 9th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'05, pages 301–311, Berlin, Heidelberg, 2005. Springer-Verlag.

[50] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, pages 412–418. ACL, 2004.

[51] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: capturing favorability using natural language processing. In *Proceedings of the*

2nd international conference on Knowledge capture, K-CAP '03, pages 70–77, New York, NY, USA, 2003. ACM.

[52] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, May 2000.

[53] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[54] Bo Pang and Lillian Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[55] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, January 2008.

[56] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[57] Owen Phelan, Kevin McCarthy, and Barry Smyth. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 385–388, New York, NY, USA, 2009. ACM.

[58] Sandi Pohorec and Milan Zorman. Formalization of unstructured content to semantic form. In Johann Eder, Mária Bieliková, and A Min Tjoa, editors, *ADBIS (2)*, volume 789 of *CEUR Workshop Proceedings*, pages 65–74. CEUR-WS.org, 2011.

[59] M. F. Porter. Readings in information retrieval, 1997.

[60] Rudy Prabowo and Mike Thelwall. Sentiment analysis: A combined approach. *J. Informetrics*, 3(2):143–157, 2009.

[61] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.*, 37(1):9–27, March 2011.

[62] Daniel Ramage, Susan Dumais, and Dan Liebling. Characterizing microblogs with topic models. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*. AAAI, 2010.

[63] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging, 1996.

[64] Jonathon Read. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, ACLstudent '05, pages 43–48, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[65] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, pages 105–112, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[66] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 172–180, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[67] Tom Rowlands, David Hawking, and Ramesh Sankaranarayana. Newweb search with microblog annotations. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1293–1296, New York, NY, USA, 2010. ACM.

[68] Hassan Saif, Yulan He, and Harith Alani. Alleviating data sparsity for twitter sentiment analysis. In *Making Sense of Microposts (#MSM2012)*, pages 2–9, 2012.

[69] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 851–860, New York, NY, USA, 2010. ACM.

[70] And Fabrizio Sebastiani, Andrea Esuli, and Fabrizio Sebastiani. Determining term subjectivity and term orientation for opinion mining andrea esuli. In *In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL '06)*, 2006.

[71] Vikas Sindhwani and Prem Melville. Document-word co-regularization for semi-supervised sentiment analysis. In *ICDM*, pages 1025–1030. IEEE Computer Society, 2008.

[72] Richard Sproat, Alan W. Black, Stanley F. Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. *Computer Speech & Language*, 15(3):287–333, 2001.

[73] Hiroya Takamura, Takashi Inui, and Manabu Okumura. Latent variable models for semantic orientations of phrases. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, Trento, Italy, 2006.

[74] David Tax. *One-class classification - Concept-learning in the absence of counter-examples*. PhD thesis, TU Delft, 2001.

[75] Gong Tianxia. Processing sentiments and opinions in text: A survey.

[76] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[77] Shoko Wakamiya, Ryong Lee, and Kazutoshi Sumiya. Towards better tv viewing rates: exploiting crowd's media life logs over twitter for tv rating. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '11, pages 39:1–39:10, New York, NY, USA, 2011. ACM.

[78] Christian Wartena and Rogier Brussee. Topic detection by clustering keywords. In *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application*, DEXA '08, pages 54–58, Washington, DC, USA, 2008. IEEE Computer Society.

[79] Michael J. Welch, Uri Schonfeld, Dan He, and Junghoo Cho. Topical semantics of twitter links. In *Proceedings of the fourth ACM interna-*

tional conference on Web search and data mining, WSDM '11, pages 327–336, New York, NY, USA, 2011. ACM.

[80] Janyce Wiebe, Theresa Wilson, and Matthew Bell. Identifying collocations for recognizing opinions. In *In Proc. ACL-01 Workshop on Collocation: Computational Extraction, Analysis, and Exploitation*, pages 24–31, 2001.

[81] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Integrated scoring for spelling error correction, abbreviation expansion and case restoration in dirty text. In *Proceedings of the fifth Australasian conference on Data mining and analystics - Volume 61*, AusDM '06, pages 83–89, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[82] Chung-Hsien Wu, Ze-Jing Chuang, and Yu-Chung Lin. Emotion recognition from text using semantic labels and separable mixture models, June 2006.

[83] Min-Chul Yang, Jung-Tae Lee, and Hae-Chang Rim. Using link analysis to discover interesting messages spread across twitter. In *Workshop Proceedings of TextGraphs-7: Graph-based Methods for Natural Language Processing*, pages 15–19, Jeju, Republic of Korea, July 2012. Association for Computational Linguistics.

[84] Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, EMNLP '03, pages 129–136, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[85] Zhang Yuhang, Wang Yue, and Yang Wei. Research on data cleaning in text clustering. In *Proceedings of the 2010 International Forum on Information Technology and Applications - Volume 01*, IFITA '10, pages 305–307, Washington, DC, USA, 2010. IEEE Computer Society.