Università degli Studi di Padova
Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea Magistrale in Informatica

Tecniche di Machine Learning per la Sentiment Analysis

Machine Learning techniques for Sentiment Analysis

Studente:

Federico Ghirardelli

Relatore:

Prof. Fabio Aiolli

Co-relatore:

Ivano Lauriola

27/09/2018

Anno Accademico 2017–2018

Indice

Elenco delle figure

Introduzione

Con la sempre maggior diffusione dei social media, gli utenti, generano una grande quantità di dati in cui sono espresse le proprie idee, preferenze e recensioni. Queste opinioni online sono diventate una forma di valuta virtuale per i brand che vogliono pubblicizzare i loro prodotti, gestire e migliorare la propria immagine e reputazione.

I social network sono diventati importanti anche nel campo della politica, utilizzati come campo di dibattito da politici e cittadini, per confronti diretti e come mezzo di comunicazione tra i più rilevanti per le campagne elettorali. Numerosi politici infatti, utilizzano sempre più Twitter, uno dei social network che maggiormente permette di esprimere e diffondere in modo veloce e conciso le proprie idee.

È essenziale quindi, nell'era dei Big Data, poter comprendere le conversazioni, individuare i contenuti di rilievo e studiare le abitudini di consumo degli utenti in modo automatico.

La Sentiment Analysis consiste nella estrazione e analisi delle opinioni, sentimenti e giudizi rispetto ad un prodotto e alle sue caratteristiche, ad un servizio, una organizzazione o più generalmente un argomento. Dai primi anni 2000, è cresciuta diventando una delle aree di ricerca più attive nel campo del Natural Language Processing (NLP).

L'applicazione della sentiment analysis, non si ferma ai soli scopi di marketing o politici ma trova sempre più spazio anche nelle scienze sociali, elaborando statistiche di natura economica, nelle scienze umanistiche, come strumento di studio delle sindromi depressive oltre che come strumento di prevenzione dei crimini come

l'adescamento di minori via chat.

Non solo il testo però è una fonte utile: grazie all'ampio uso degli smartphone e quindi all'elevato uso delle fotocamere e dei messaggi vocali, sta crescendo anche l'interesse nello studio delle emozioni nei settori della image analysis e speech analysis. Questa ricerca ha prodotto numerose tecniche per l'automazione della sentiment analysis che ricadono nel campo dell'apprendimento automatico. Affinché gli algoritmi di apprendimento automatico possano processare i dati per allenarsi ed imparare a riconoscere la polarità, per poi poter classificare esempi mai visti prima, è necessario estrarre quelle che sono le caratteristiche (*fatures*), cioè le proprietà dei dati osservati.

Questo lavoro di tesi mira a fornire una panoramica dello stato dell'arte della Sentiment Analysis, studiare e confrontare le rappresentazioni delle feature testuali e a implementare diversi modelli di Machine Learning in grado di classificare testi in base alla polarità espressa. L'idea è quella che analizzando brevi testi o frasi, sia possibile determinare se l'opinione generale espressa dall'autore è positiva, negativa oppure neutra. Verranno utilizzati come benchmark alcuni dataset ormai noti nell'ambiente accademico, riguardanti soprattutto i tweet e le recensioni di film, dedicando particolare attenzione al preprocessing dei dati testuali, alla estrazione e alla rappresentazione delle features da utilizzare nei diversi modelli di apprendimento. Tra le varie tecniche di apprendimento automatico, verrà analizzata maggiormente la tecnica del Deep Learning, le varie tipologie di rete neurale artificiale e le componenti che vengono utilizzate per creare topologie sempre più complesse, come le reti convoluzionali a più livelli e i modelli concatenati, architetture che basate su più sottoreti distinte che combinano tipi di rappresentazione diverse.

La tesi è così strutturata:

Nel primo capitolo, **Sentiment Analysis** viene presentata la panoramica della Sentiment Analysis, le definizioni più importanti di questo dominio ed un rapido riassunto delle applicazioni e delle sfide principali che ne derivano. Sono descritti i diversi livelli di granularità a cui la sentiment analysis è comunemente applicata.
Lo studio si è focalizzato sulla **Estrazione delle features** dei dati testuali, analizzando e confrontando sia le caratteristiche prettamente legate al linguaggio naturale che quelle di natura prettamente statistica e di più alto livello.
In **Tecniche di classificazione** vengono schematizzate le principali tecniche di classificazione della polarità e vengono approfondite le tecniche di machine learning.
Deep Learning introduce le reti neurali artificiali, le diverte tipologie di rete, soffermandosi maggiormente sulle reti neurali profonde e le componenti dei singoli strati.
Implementazione presenta l'ambiente di sviluppo, i dataset e le loro caratterstiche, gli algoritmi di preprocessing e pulizia dei dati ed infine i modelli di machine learning utilizzati.
Nel capitolo **Risultati** vengono riportati ed saminati i risultati conseguiti con i diversi sistemi implementati, riportando i modelli e le configurazioni delle feature che hanno riportato i valori più interessanti delle metriche principali.
La tesi termina con l'ultimo capitolo **Conclusioni**, discutendo in modo generale il lavoro conseguito ed evidenziando le criticità incontrate, i possibili miglioramenti e sviluppi futuri.

Capitolo 1

Sentiment Analysis

In questo capitolo viene presentata la panoramica riguardante la Sentiment Analysis, le metodologie a cui è legata e le principali definizioni utili per i capitoli a venire.

1.1 Analisi Testuale

Per introdurre la Sentiment Analysis, è utile descrivere alcune delle più importanti discipline che ancora prima di integrare lo studio dei sentimenti, hanno formulato le metodologie e gli algoritmi utili a rendere un testo prima di tutto, decodificabile e analizzabile in modo automatico.

Natural Language Processing

L'analisi del sentimento sta ricevendo un interesse sempre crescente dalla comunità del *Natural Language Processing* (NLP), particolarmente motivata dalla diffusa necessità di applicazioni basate sulle opinioni, come recensioni di prodotti, analisi delle caratteristiche dei prodotti e sul riepilogo dei sentimenti.
NLP è un campo di ricerca basato su tecniche di psicologia cognitiva e di analisi linguistica che, con l'aiuto dei calcolatori e di algoritmi di apprendimento, permette di analizzare e decodificare un testo. L'idea è quella di modellare, attraverso l'uso di algoritmi, il modo in cui si forma il linguaggio umano. I metodi NLP sono di tipo

più esplorativo delle struttura dei testi e delle relazioni tra le parole e i contenuti piuttosto che finalizzate alla stima delle opinioni.

La Linguistica Computazionale (*Computational Linguistics - CL*) è la base scientifica su cui poggia il NLP. Possiamo quindi dire che NLP ne è la sua applicazione.

Altro termine tecnico è *Natural Language Understanding* (NLU), utilizzato per indicare solitamente una analisi più approfondita di tipo morfologico del linguaggio naturale. Il linguaggio naturale ha due forme, parlato (forma orale) e scritto (forma scritta).

In questo lavoro di testi, i dati presi in esame sono interamente testuali.

La maggior parte delle tecniche di NLP ereditano in gran parte dalla Linguistica e dall'Intelligenza Artificiale. Le metodologie sono ancor più recentemente interessate da domini come la statistica computazionale, la Scienza cognitiva e come vedremo in questo lavoro di tesi, dall'Apprendimento automatico.

Information Retrival

Ancora prima della analisi e la classificazione del testo, vi è una questione cruciale a fini del text mining, quella del reperimento dei dati stessi.
L'*Information Retrieval* (IR) è l'insieme delle tecniche che gestiscono la rappresentazione, la memorizzazione, l'organizzazione e l'accesso a documenti. Permette di ritrovare documenti e informazioni rilevanti in una collezione di documenti, in relazione a specifiche domande e esigenze dell'utente.

Information extraction

Il compito della *Information Extraction* (IE), è quello di estrarre informazioni specifiche da documenti non strutturati (come il testo puro) e/o documenti semi-strutturati (ad esempio XML). Le sue tecniche si basano sulle ricorrenze di termini e frasi all'interno dei documenti e prevedono metodi di *pattern matching* e *machine learning*. L'obiettivo non è tanto quello di estrarre l'opinione ma piuttosto di estrarre i concetti che descrivono il contenuto di un documento e avere una rappresentazione strutturata di essi. Le sue operazioni sono complementari al IR: data una base

documentale ed una query, l'IR restituisce un sottoinsieme di documenti rilevanti per la query; l'IE estrae in modo strutturato le informazioni rilevanti, pronte per le fasi *postprocessing*.

Topic detection and Tracking

Poiché la rete ha una quantità eccessiva di informazioni e le informazioni relative ad un argomento specifico sono spesso disperse, tra fonti di nazioni diverse con orari diversi da parte di siti di news o di broadcasting, può essere difficile reperire e comprendere in modo chiaro e compatto un argomento.
Per aiutare le persone a integrare informazioni sparse, la *Topic detection and Tracking* (TDT), racchiude tutte quelle indagini utili ad identificare e monitorare l'utilizzo di keyword nella rete e all'intero di numerosi corpus di documenti [2].

Text Summerization

Cerca di trasformare un testo in uno più breve che lo riassuma. Fa uso di tecniche statistiche per valutare le frequenza delle parole o delle frasi. Un esempio classico derivante da questa tecnica è il *tag cloud*.

1.2 Analisi del sentimento

Le tecniche presentate precedentemente, sono state sviluppate principalmente per problemi di topic detection. Più recentemente, gli studi hanno iniziato a dare un peso sempre maggiore agli autori dei tesi, ai soggetti a cui gli autori si riferiscono e in particolare alle opinioni che tali autori rivolgono. Da questi studi sono nate la *sentiment analysis* e l'*opinion mining*. Entrambe rientrano nel contesto più generale dell'elaborazione del linguaggio naturale, della linguistica computazionale e dell'analisi testuale.

L'**opinion mining** si occupa di estrarre, in modo automatico, opinioni, sentimenti ed emozioni contenute in un documento testuale, sia esso un articolo di giornale, una recensione, un commento o un post su social network, blog o forum.

L'obiettivo della **sentiment analysis** è quello di identificare e classificare le opinioni espresse su un determinato soggetto, o argomento, o, più in generale, determinare la polarità (ad esempio positiva, negativa o neutra) che l'autore dell'opinione esprime.

Una **Opinione** è definita come: Idea, parere, giudizio relativo a qualcuno o qualcosa;
Ci sono due tipi di opinioni: *opinioni regolari* e *pareri comparativi*.
Una opinione regolare esprime un giudizio unicamente sull'entità, sul soggetto indicato. Questa può essere a sua volta, una opinione *oggettiva* oppure *soggettiva*.
I pareri comparativi, mettono in relazione uno o più soggetti, comparandone caratteristiche che li distinguono.

Con **Sentimento** si intende ogni stato affettivo della coscienza, di segno positivo o negativo; ogni moto soggettivo dell'animo che dia una particolare tonalità affettiva alle nostre sensazioni, rappresentazioni, idee.

Una definizione formale di opinione è stata fornita da Liu, Zang (2012) [1]. Essi rappresentano una opinione come una quintupla
opinione = (e_i, a_{ij}, s_{ijkl}, h_k, t_l)

ove:

- $\mathbf{e_i}$ rappresenta un'entità cioè l'oggetto su cui si sta esprimendo l'opinione, sia esso una persona, un evento, un argomento, un prodotto. L'entità può avere diverse proprietà che la caratterizzano ed essere composta da altre componenti (sotto-entità) che, a loro volta, possono avere proprietà e componenti. Formalmente, quindi, ad ogni entità può esser associata una coppia (C, A) dove C è l'insieme di tutti i suoi componenti mentre A è l'insieme di tutte le sue proprietà;
- $\mathbf{a_{ij}}$ è la proprietà (o componente) j dell'entità e_i, rispetto alla quale è espressa l'opinione;
- $\mathbf{h_k}$ è l'autore dell'opinione;

- $\mathbf{t_l}$ è l'istante temporale in cui h_k ha espresso la propria opinione;
- $\mathbf{s_{ijkl}}$ rappresenta l'orientamento (polarità); dell'opinione sulla proprietà a_{ij} dell'entità e_i al tempo t_l da parte di h_k. Tale orientamento può essere positivo o negativo e può avere diversi livelli di intensità.

1.2.1 Challenges

La letteratura, ad oggi, si è concentrata particolarmente sui seguenti problemi di classificazione legati alla sentiment analysis:

- Estrazione delle entità (*Entity extraction*)
- Estrazione delle proprietà soggette ad opinione (*Aspect Extraction*)
- Estrazione dell'autore dell'opinione (*Opinion holder extraction*)
- Estrazione e standardizzazione del tempo (*Time Extraction and Standardization*)
- Classificazione delle proprietà (*Aspect Sentiment Classification*)
- Generazione quintuple di opinione (*Opinion Quintuple Generation*)
- Soggettività/oggettività (*Subjectivity classification*)
- Polarità di testi soggettivi (*Sentiment polarity classification*)

La comunità scientifica è oltremodo spinta sempre più dai fattori economici, soprattutto dai brand, che vogliono migliorare le proprie strategie di marketing, sfruttando appieno le risorse che i social media mettono a disposizione. Numerosi sono i tool nati negli ultimi anni, dedicati principalmente ai data scientist, per il reperimento automatico attraverso API gratuite e non, dei post degli utenti Twitter e Facebook. Possiamo chiamare questa tecnica *Social Listening*.

Marketing

Attraverso il Social Listening, ne traggono enormi vantaggi le aziende e i marchi che vogliono migliorare la propria *Customer Experience*, monitorando i feedback post vendita. Primi fra tutti gli Hotel, che "ascoltando" le critiche dei clienti, cercano di capire quali sono i propri punti deboli e dove possono intervenire per fornire un servizio sempre migliore.
Molte sono le catene che hanno la necessità di accertarsi che il livello di qualità sia lo stesso per ogni loro sede, indipendentemente dal posizionamento geografico.

Interpretando le recensioni, si può operare in real-time laddove vi siano post ad alto rischio, fornendo assistenza nel minor tempo possibile, minimizzando le ulteriori critiche degli utenti che si rispecchiano nella critica originale. Si può analizzare la reazione del pubblico alla sponsorizzazione, monitorando le differenze tra le diverse categorie di pubblico, ad esempio in base ai diversi fattori sociali, etnici, religiosi o politici. Nelle prime fasi di lancio di un prodotto, è possibile sfruttare la sentiment analysis per capire come vengono percepite le nuove feature (intese come caratteristiche e funzionalità del prodotto) e laddove vi siano forti criticità, preparare al più presto una strategia di assistenza tecnica, una strategia di correzione di eventuali bug critici se ad esempio si tratta di un prodotto software o di sostituzione e rimborso. Altrettanto importante è il dover monitorare la popolarità dell'immagine aziendale, dei membri dirigenti, primo fra tutti il CEO, che sono figure chiave per l'immagine pubblica.

Politica

La sentiment analysis nell'ambito del microblogging ha mostrato come i social possano essere un valido indicatore del sentiment e del trend politico. I tweet ad esempio, possono fornire una panoramica che riflette il panorama politico e statistiche in tempo reale.

Opinion Spam Detection

Tra le numerose recensioni, non mancano quelle false, chiamate anche *bogus reviews*, o con il fine di danneggiare la reputazione di un brand, ingannare e truffare altri utenti, promuovere entità esterne dal contesto.
Compito dell'*Opinion Spam Detection* è di rilevare in modo automatico questo tipo di recensioni.

Oltre al problema della polarità, ci sono più livelli di significato nelle frasi generate dall'uomo. Le persone esprimono opinioni in modi complessi; facendo uso di retorica e sarcasmo. L'ironia e il significato implicito possono trarre in inganno l'analisi del sentimento. Per studiare questi livelli linguistici più complicati e fini, vi sono diversi altri sotto-task:

Sarcasm Analysis

Il sarcasmo è una figura retorica utilizzata dalle persone per fare un particolare tipo di ironia ed esprimere giudizi con l'intento di umiliare o sotto intendere altri concetti. Mentre si parla, le persone spesso usano uno particolare tono di voce e determinati indizi gestuali come il rotolamento degli occhi, il movimento delle mani, ecc. Per rivelare il sarcasmo. Nei dati testuali, questi indizi tonali e gestuali mancano, rendendo la rilevazione del sarcasmo molto difficile per un umano medio. Gli unici indizi utili possono essere l'uso delle virgolette ("") e nella comunicazione online l'aggiunta delle emoticon, per dare una valenza particolare alla frase. A causa di queste sfide, i ricercatori mostrano interesse per il rilevamento del sarcasmo del testo dei social media, specialmente nei tweet.

Emoticons

Strettamente legati allo studio del sarcasmo, diversi sono gli studi che ruotano attorno alle emoticons, sempre più utilizzate nei contesti online e in grande quantità all'interno dei singoli messaggi, quasi a formare una sorta di nuova lingua. Questi studi, cercando di capire quali sono le emozioni associate ad ognuna delle emoticons.

È stato riscontrato, che le emozioni per la stessa emoticon possono risultare contrarie e discordanti a seconda della nazionalità o religione dell'autore [15].

1.3 Livelli di applicazione

La ricerca ha studiato principalmente la sentiment analysis su alcuni livelli di granularità del testo [28], così come rappresentato in figura 1.1.
Le classificazioni su livelli diversi non sono indipendenti l'una dall'altra ma, anzi, ognuna di esse dipende dall'analisi di granularità inferiore: le classificazioni a livello di documento dipendono fortemente da quelle di frase che, a loro volta, dipendono da quelle a livello di parola.

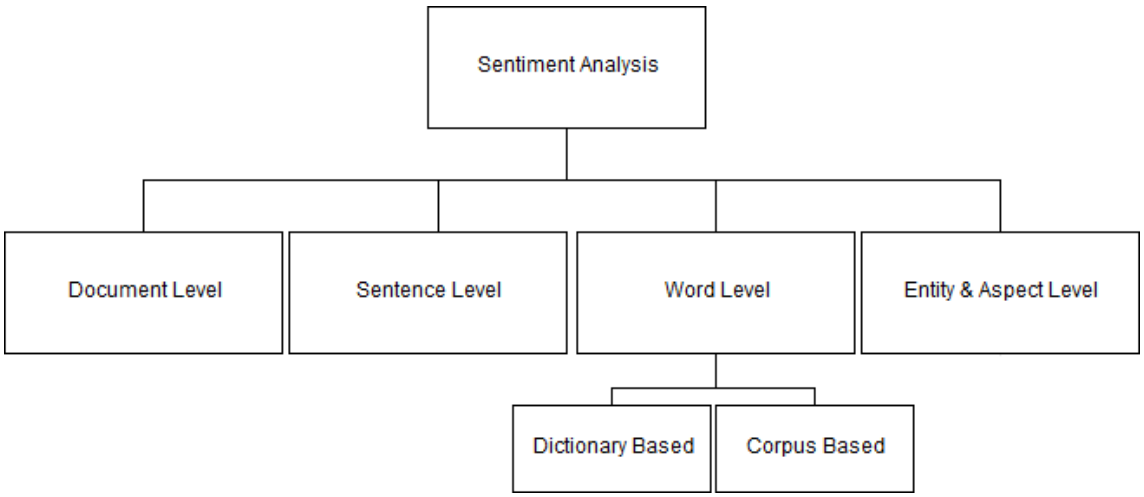


Figura 1.1: Livelli di granularità

A livello di documento (*Document Level*), il compito principale è quello di classificare se un intero documento di opinione esprime un sentimento positivo o negativo. Questo livello di analisi presuppone che ogni documento esprima opinioni su una singola entità. A livello di frase (*Sentence Level*), il compito principale è verificare se ogni frase esprime un'opinione positiva, negativa o neutra. Il *livello di aspetto* esegue analisi a grana più fine. Invece di guardare i costrutti linguistici (documenti, paragrafi o frasi), guarda direttamente alle opinioni, cercando prima di tutto, di identificare quali sono i soggetti dell'opinione, le entità e le caratteristiche

di esse menzionate. *Word Level* analizza le parole prese singolarmente all'interno del testo.

1.3.1 Document Level

Dato un documento di opinione che valuta una entità, si vuole determinare il sentiment complessivo dell'autore rispetto all'entità, cioè si vuole determinare s riguardo un aspetto GENERAL nella quintupla *(_,GENERAL,s,_,_)*, presupposto che l'entità e, l'opinion holder h e il tempo t siano conosciuti.
Se s assume valori categoriali che descrivono la polarità (positiva, negativa, neutra), si parla di un problema di *classificazione*.
Se s assume valori numerici o punteggi ordinali all'interno di un determinato intervallo (esempio da 1 a 5), si parla di un problema di *regressione*.

1.3.2 Sentence Level

Per la classificazione del sentiment a livello di documento, è prima di tutto importante scendere di un livello di granularità, cioè adoperare la corretta classificazione delle frasi che compongono il documento.

Nel Sentence Level ci si occupa del riconoscimento e della classificazione di singole frasi o di brevi messaggi di testo (*short-text*).
Questo livello di analisi è strettamente correlato alla classificazione della soggettività [5], che distingue frasi oggettive che esprimono informazioni fattuali da frasi soggettive che esprimono opinioni soggettive.
Bisogna sottolineare che la soggettività non è equivalente al sentimento in quanto molte frasi obiettive possono implicare opinioni la cui verità è "assoluta" e non propriamente utile al fine della sentiment analysis.

1.3.3 Word Level

Molte delle metodologie adottate per la sentiment analysis a livelli di frase utilizzano la polarità a priori delle singole parole contenute nella frase stessa.

Il valore di polarità viene solitamente ricavato accedendo ad un dizionario di *opinion word* (detto **lexicon**), che può essere costruito con tecniche manuali oppure con tecniche automatiche, ed opportunamente etichettato.
La creazione manuale di un lexicon prevede la selezione e la classificazione di aggettivi, nomi, verbi e avverbi a partire da un tradizionale dizionario.

1.3.4 Aspect Level

Chiamato in precedenza *Feature Level*, mira direttamente ad analizzare l'opinione di una entità. Si basa sull'idea che un'opinione consiste di un sentimento (positivo o negativo) e di un obiettivo (di opinione).
Come già menzionato, un soggetto può essere solitamente una entità o una proprietà dell'entità stessa. Per esempio, nella frase *"la fotocamera è ottima ma la durabilità della batteria è pessima"*, l'entità è una sola, ad esempio uno smartphone, ma le proprietà descritte sono due: la sua fotocamera, a cui l'utente esprime un giudizio positivo, e la batteria, sui si esprime negativamente.
È ovvio come può risultare complicato analizzare il sentimento generale dell'intera frase, perché formata da due diversi giudizi e soprattutto di polarità contraria.

Capitolo 2

Features extraction

I dati di addestramento richiedono quasi sempre, una prima fase di pre-processing, al fine di essere espressi come un insieme di "features", ridurre la dimensionalità dei dati stessi per una elaborazione più veloce e "pulire" o correggere parti del testo che non sono portatici di significato utile all'analisi del sentimento. Vi sono fasi in cui i dati vengono invece arricchiti e la dimensione dello spazio delle feature aumentato.

In questa sezione vengono esposte le principali feature che vengono estratte comunemente dal testo e le metodologie utilizzate.

2.1 Pre-processing

Per avviare una analisi testuale, è necessario prima di tutto poter decodificare ed interpretare il testo in maniera automatica, affinché gli algoritmi di machine learning possano allenarsi. Questa fase preliminare è chiama *pre-processing* del testo.

2.1.1 Tokenizzazione

Il primo passaggio necessario è la riduzione del testo in un dato quantitativo. La fase di *tokenizzazione* si occupa di suddividere il testo in unità elementari, i *token*: non solamente le singole parole, ma anche la punteggiatura. Tale suddivisione, come si vedrà nella fase sperimentale non è univoca ma dipende fortemente dal contesto e per tale scopo esistono diversi *tokenizer*.

2.1.2 Pulizia e riduzione della dimensionalità

Esistono tecniche per la riduzione delle *features* che eliminano parte dei termini che compongono i vettori di features. L'idea è di eliminare i termini meno significativi dall'insieme totale. La tecniche più utilizzate, valutano la *sparsity* di un insieme di features. Una matrice si dice *sparsa* quando la maggior parte dei suoi valori sono zeri, al contrario delle matrici con un alto tasso di valori non-zero, chiamate *dense*. Rappresentare e lavorare con matrisi sparse è computazionalmente costoso. Si possono ottenere miglioramenti nelle prestazioni utilizzando rappresentazioni che riducano la sparsità. L'idea più comune è valutare la frequenza documentale dei termini. Ad esempio, nella costruzione di un vocabolario di parole, utilizzato per la costruzione delle già descitte Bag-of-word, vengono scartate le parole aventi una frequenza sotto una certa soglia. Una alternativa, largamente studiata nell'Information Retrieval, è utilizzare la *legge di Zipf*, dove sono date due soglie, una di upper bownd, e una di lower bound, secondo l'idea che, così come le parole poco frequenti siano portatrici di scarso significato, lo siano anche quelle troppo frequenti, perchè utilizzate in modo eccessivo. Tali soglie, sono anch'esse soggette alla fase di tuning dei parametri, per trovare il giusto trade-off tra la dimensione del numero di feature e il costo computazionale.

2.1.3 Stemming

Sebbene si possa rappresentare un documento con le parole grezze in esso contenute, una tecnica classica dell'Information Retrivale, per ridurre la dimensionalità del vocabolario per l'indicizzazione, è quella di trasformare le parole nella loro forma flessa e più generale, alle loro radici morfologiche. Tale tecnica prende il nome di *Stemming* e le parole trasformate vengono chiamate *stem*.
Ad esempio, le parole "computer", "computational" e "computing" sono ridotte alla radice "comput".

La riduzione a radice può avvenire con diversi gradi di stem. Aumentando il grado di stem la radice diventa più corta e si amplia il numero dei termi nella famiglia di

parole che identifica. Le regole di stemming cambiano con la lingua.

Come dimostrano Kushal et al. [21], nell'ambito della sentiment analysis, "troncare" le parole, può produrre risultati misti, a seconda del dominio. Riducendo una parola alla propria radice, si rischia molto spesso di perdere il significato semantico della parola stessa all'interno della frase. La parola stemmata, ad esempio, non è più portatrice del tempo verbale originario. Analizzando recensioni di prodotti, osservano che le recensioni negative tendono a verificarsi più frequentemente al passato, poiché ad esempio, il prodotto potrebbe esser stato restituito.

2.1.4 Stopword

Per ridurre ulteriormente la complessità del testo, si possono eliminare quelle che vengono chiamate *stopword*. Le stop-word sono quelle parole che non hanno un significato specifico, si tratta di parole comuni usate in ogni testo (ad esempio articoli, preposizioni, negazioni ecc.).

2.1.5 Duplicati

In alcuni dataset può presentarsi più volte lo stesso dato, introdotto per errore o perchè non riconosciuto come duplicato, come ad esempio frasi dallo stesso contenuto ma con punteggiatura o codifica dei caratteri diversa. La seconda opzione, può rendere più difficile riconoscere ed eliminare i duplicati. Si potrebbe erroneamente assumere che il numero di questi dati sia esiguo. Nei dataset aventi come dominio i tweet, sono state riscontrate percentuali maggiori di duplicati di tweet positivi rispetto a quelli in tweet negativi perché le persone tendono a retwittare qualcosa di cui sono contenti.

2.1.6 Correzione grammaticale

In molti contesti, come nei tweet, una correzione grammaticale è pressoché impossibile, dato l'alto tasso di parole storpiate o di recente introduzione.

Una possibile correzione è quella che mira a ridurre il numero di caratteri ripetuti all'interno di alcune parole. Molte delle parole che esprimono sentimenti positivi, vengono twittate con una delle lettere centrali o finali ripetute, ad esempio *"happyyyyyy"* può essere corretta alla sua forma corretta *"happy"*.

2.2 Features representation

Con *feature extraction* si intende il procedimento che trasforma un testo in un vettore di feature numeriche. Diverse sono le caratteristiche estraibili dal testo e diversi i modi di rappresentarle. Di seguito vengono descritte le rappresetnazioni principali e più utilizzate nel campo del NLP.

2.2.1 Bag-of-Words

Il modello *Bag-of-words* (BoW - in italiano tradotto come *Modello della borsa di parole*), è un metodo per rappresentare il testo ignorando l'ordine della parole.

Bow richiede:

- Un vocabolario delle parole contenute nel testo in esame;
- La frequenza di tali parole.

È chiamato "bag" (borsa) perchè non mantiene alcuna informazione sulla struttura o l'ordine delle parole. Il modello riguarda solo le parole conosciute che verificano nel documento, non dove. L'intuizione è che i documenti sono simili se hanno contenuti simili. Dal solo contenuto, possiamo imparare qualcosa sul significato del documento.

L'approccio utilizzato in questo lavoro può essere così riassunto:
Considerato un insieme di frasi:

"Nel mezzo del cammin"
"Quel ramo del lago di Como"

viene creato un vocabolario. Dalle frasi si ottengono i token, le singole parole, prima ridotte a caratteri minuscoli.

"nel" "mezzo" "del" "cammin" "quel" "ramo" "lago" "di" "como"

Il passo successivo è creare i vettori di features utilizzabili dagli algoritmi di machine learning. Il processo di conversione del testo in vettori di numeri è chiamato *vectorization*. Gli approcci si differenziano a seconda che si tenga conto della frequenza delle parole oppure della semplice binarietà (uni e zeri). Per frasi molto corte, come nel nostro caso, la binarietà è risultata essere sufficente, oltre che computazionalmente più efficente.

In questo caso, le parole totali sono 10. Il vocabolario, formato delle parole uniche, ha dimensione pari ad 9. Ogni frase viene così rappresentata:

"nel mezzo del cammin": $[1, 1, 1, 1, 0, 0, 0, 0]$
"quel ramo del lago di como": $[0, 0, 1, 0, 1, 1, 1, 1, 1]$

Limitazioni

Bow risulta essere uno dei modelli di feature extraction più semplici e intuitivi, largamente utilizzato in problemi di classificazione di documenti e spam filtering. Tuttavia soffre di alcune carenze e richiede una attenta progettazione del vocabolario. Bisogna tenere conto della dimensione e quindi della sparsità dei vettori di features che rappresentano i documenti, o come in questo lavoro, le singole frasi.

2.2.2 TfIdf

La funzione di peso **tf-idf** (*term frequency-inverse document frequency*) è una funzione utilizzata in information retrieval per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti. Tale funzione aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento, ma cresce in maniera inversamente proporzionale con la frequenza del termine nella

collezione. L'idea alla base di questo comportamento è di dare più importanza ai termini che compaiono nel documento, ma che in generale sono poco frequenti.

La funzione può essere scomposta in due fattori: Il primo fattore della funzione è il numero dei termini presenti nel documento. In genere questo numero viene diviso per la lunghezza del documento stesso per evitare che siano privilegiati i documenti più lunghi.

$$tf_{i,j} = \frac{n_{i,j}}{|d_j|} \tag{2.1}$$

dove $n_{i,j}$ è il numero di occorrenze del termine i nel documento j, mentre il denominatore $|d_j|$ è la dimensione, espressa in numero di termini, del documento j. L'altro fattore della funzione indica l'importanza generale del termine i nella collezione:

$$idf_i = log\frac{|D|}{|\{d : i \in d\}|} \tag{2.2}$$

Dove $|D|$ è il numero di documenti nella collezione, mentre il denominatore è il numero di documenti che contengono il termine i.
Abbiamo quindi che:

$$(tf - idf)_{i,j} = tf_{i,j} \times idf_i \tag{2.3}$$

2.2.3 Word embedding

Il *word embedding* è una classe di approcci per rappresentare parole e documenti utilizzando una rappresentazione vettoriale densa, secondo l'idea che parole con significato simile, assumano una rappresentazione nello spazio vettoriale simile.

"Uno dei vantaggi dell'utilizzo di vettori densi e di bassa dimensione è computazionale: la maggior parte dei toolkit di reti neurali non funziona bene con vettori molto sparsi. Il principale vantaggio delle rappresentazioni dense è il potere di generalizzare: se riteniamo che alcune caratteristiche possano fornire indizi simili, è utile fornire una rappresentazione che sia in grado di cogliere queste somiglianze." [24]

Ogni parola viene mappata su di un vettore e i suoi valori vettoriali vengono appresi in un modo simile ad una rete neurale. Per questo, questa tecnica, viene spesso inserita nel campo del deep learning.

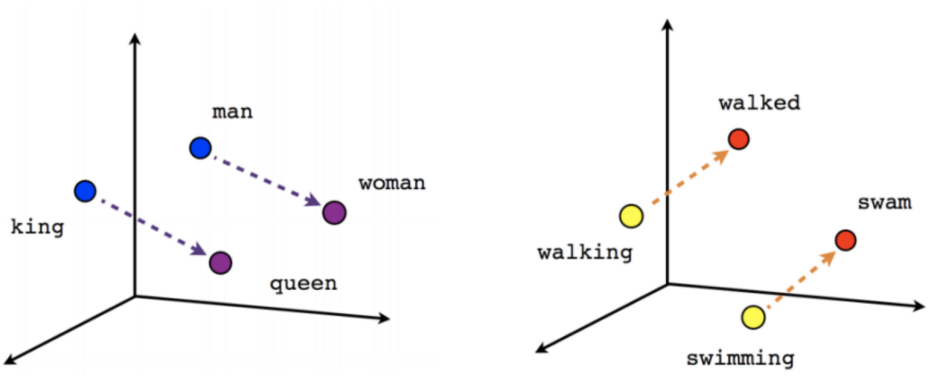


Figura 2.1: Esempio di word embedding

La rappresentazione viene appresa in base all'uso delle parole. Ciò consente alle parole utilizzate in modo simile di ottenere rappresentazioni simili, in contrasto con le rappresentazioni sopra descritte, come il modello Bow,dove diverse parole hanno rappresentazioni diverse, indipendentemente dal modo in cui vengono utilizzate.
I metodi embedding delle parole apprendono una rappresentazione vettoriale a valori reali per un vocabolario predefinito di dimensioni fisse a partire da un corpus di testi.

Gli algoritmi più utilizzati per il word embedding sono:

- **Keras Embedding Layer**

  Keras offre un livello di Embedding che può essere utilizzato come livello per le reti neurali. Richiede che i dati di input siano codificati come numeri interi, in modo che ogni parola sia rappresentata da un numero intero univoco. Il livello di Embedding è inizializzato con pesi casuali e imparerà l'embedding per tutte le parole nel set di dati di addestramento.

  Può essere utilizzato in vari modi:
  Può essere usato da solo per imparare un embedding delle parole da riutilizzare poi in modello in futuro; come parte di un modello di deep learning in cui

l'embedding viene appreso insieme al modello stesso; per caricare un modello di embedding di parole ottenuto in un pre-training, per task di *transfer learning*.

È definito come il primo livello nascosto di una rete neurale e deve specificare 3 argomenti:
- *input_dim*: dimensione del vocabolario. Se ad esempio il vocabolario predefinito ha dimensione 10, le parole dovranno essere codificate con valori da 0 a 9.
- *output_dim*: dimensione dello spazio vettoriale in cui verranno rappresentate le parole. Definisce la dimensione dei vettori di output da questo livello per ogni parola.
- *input_length*: lunghezza di vettori di input. Se ad esempio le frasi sono formate da 500 parole, i vettori di input dovranno avere tutti la dimensione pari a 500.

- **Word2Vec**

  Word2Vec è un metodo statistico per l'apprendimento dell'embedding di parole contenute in un corpus di testi.

  Sviluppato da Tomas Mikolov, et al. a Google nel 2013, per rendere più efficiente l'addestramento di embedding basato reti neurali. Il vantaggio principale dell'approccio è che gli embeddings delle parole possono essere appresi in modo efficiente (con uno spazio ridotto e una ridotta complessità temporale), consentendo di apprendere corpus di testo di grandi dimensioni.

- **GloVe**

  Global Vectors for Word Representation (GloVe), è un algoritmo ad estensione del metodo Word2Vec sviluppato da Pennington, et al. a Stanford.

  Le rappresentazioni delle parole sono state sviluppate usando tecniche di fattorizzazione di matrici come la *Latent Semantic Analysis (LSA)* che utilizza le statistiche globali del testo.
  Anziché utilizzare una finestra per definire il contesto locale, GloVe costruisce

una matrice esplicita di parole-contesto o matrice di co-occorrenza delle parole utilizzando le statistiche dell'intero corpus di testo.

2.3 Metodi per estendere il vocabolario

2.3.1 Part-of-speech tags

Part-of-speech tagging, è il processo di disambiguazione di categorie di parole, che attribuisce ad ogni parola in un testo (o in un corpus di testi), un tag corrispondente ad una particolare parte del discorso, basato sia sulla definizione della parola stessa, sia sul contesto e sulla relazione che la parola ha con le altre nella frase in cui occorre.

In inglese per esempio, POS tag comuni sono: NN(sostantivo), VB(verbo), JJ(aggettivo), AT(articolo), VBZ(verbo, terza persona singolare), PPN(nome proprio singolare), PRP(pronome personale).

Etichettare le caratteristiche grammaticali di ogni parola è anche un'ottima strategia per rilevare schemi utili per le classificazioni. I testi soggettivi ad esempio, tendono a usare il passato semplice invece del participio passato. Al contrario delle frasi positive, quelle negative contengono più spesso verbi al passato, perché, esempio classico delle recensioni di prodotti, gli autori esprimono i loro sentimenti negativi riguardo alla delusione che hanno provato.

Gli algoritmi di POS tagging si divisono in due categorie:

- *Rule-based POS-tagger*: gli approcci basati su regole utilizzano informazioni contestuali per assegnare tag a parole sconosciute o ambigue. La disambiguazione è fatta analizzando le caratteristiche linguistiche della parola, la sua parola precedente, la sua parola successiva ed altri aspetti. Ad esempio, se la parola precedente è un articolo, la parola in questione dovrà essere un nome. Queste informazioni sono codificate sotto forma di regole.

- *Stochastic POS-tagger*: I tagger stocastici disambigono le parole basandosi unicamente sulla probabilità che una parola si verifichi con un determinato tag. L'idea è che il tag incontrato più frequentemente nel set di allenamento con una parola, è quello assegnato a una istanza ambigua di quella parola.

Sebbene i tool per il pos tagging abbiano raggiunto un ampio consenso nel campo del NLP, rimangono diversi casi in cui risulta molto difficile atrribuire corretamente pos tag alle parole. Alcuni tool, forniscono, per la stessa parola, la lista di possibili tag qual'ora vi sia incertezza.

2.3.2 Sentiment polarity shifters

Vi sono alcuni fenomeni linguistici che possono causare l'inversione di polarità di una parola o l'aumento della sua intensità semantica. [23]

- **Negazioni**

  Una parte particolarmente utile del discorso per il compito di determinare la classe di polarità è lo studio delle negazioni. Le negazioni sono tutte quelle parole o modi di dire che invertono la polarità della parola a cui sono applicate o delle frasi succedono ad esse.

  Un esempio di negazioni sono le parole *not* e *never*.

  Spesso vengono eliminate dal testo se si adopera la tecnica di stemming precedentemente descritta, in quanto riconosciute come stopword.

  L'approccio di Das et al. [26] usa un'euristica per identificare le negazioni e creare una nuova feature aggiungendo NOT alle parole (ad esempio, la frase *"don't like"* viene tradotta nella feature NOT-like).

- **Intensificatori**

  Gli intesificatori, sono termini che possono sia amplificare che diminuire la polarità delle parole che accompagnano. Ad esempio, la parola "very", nel caso in cui preceda "happy", risulta essere un amplificatore della positività già espressa da "happy". Al contrario, se abbinata ad esempio alla parola "sad", concorre ad aumentare la polarità negativa del contesto.

2.3.3 N-grams

Le frasi di negazione discusse sopra possono essere considerate come un caso speciale di n-grammi. Il vantaggio dell'uso di n-grammi invece di singole parole come feature è la capacità di catturare alcune dipendenze tra le parole e l'importanza delle singole frasi. Gli n-grammi sono sequenze di n token di una sequence.
Un esempio di n-grammi:

Data la frase:

"Ho adorato davvero questo film"

La sequence di token in rapprestazione 1-gram e 2-gram sono):

- **1-gram (uni-grammi)**: *["ho", "adorato", "davvero", "questo", "film"]*
- **2-gram (bi-grammi)**: *["ho adorato", "adorato davvero", "davvero questo", "questo film"]*

Hang et al. [27] hanno trovato un significativo miglioramento nel compito di classificazione della polarità usando un n elevato (fino a 6).

Capitolo 3

Tecniche di classificazione

In questo capitolo vengono esposte le principali tecniche di classificazione per la sentiment analysis, secondo la suddivisione proposta in figura 3.1.

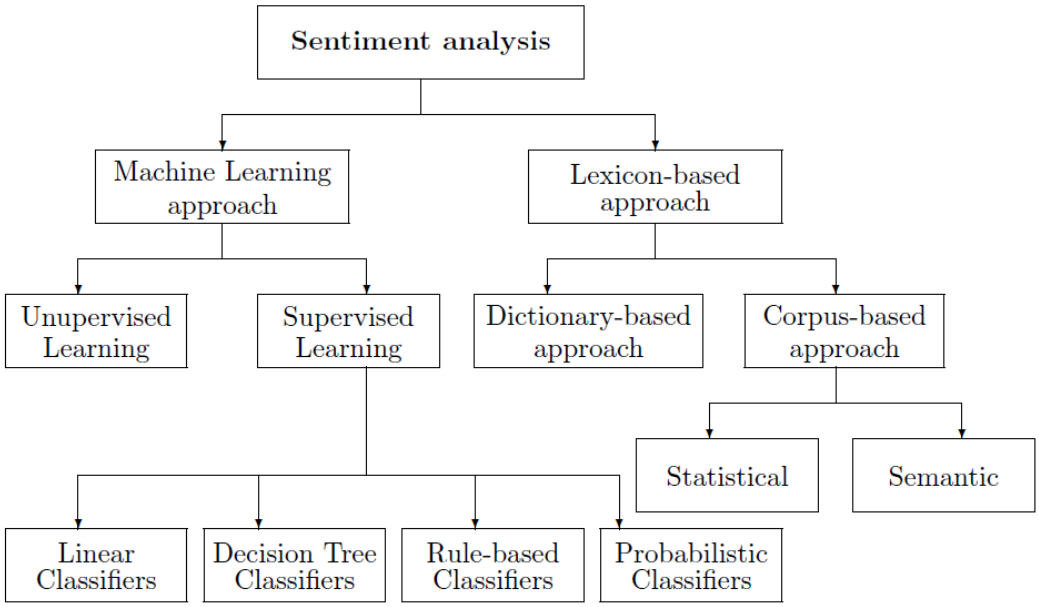


Figura 3.1: Tecniche per la classificazione della sentiment analysis

Le prime due macro-categorie riguardano gli approcci più frequenti con cui si può costruire il processo di analisi dei testi, a prescindere dal livello di granularità tra quelli descritti nel capitolo precedente: l'approccio legato all'apprendimento automatico (*Machine Learning approach*) e l'approccio legato al lessico del testo (*Lexicon-based approach*).

Va sottolineato come questi approcci possano essere utilizzati singolarmente oppure, come spesso accade nei modelli più complessi, in combinazione tra di essi.

Gli approcci basati sul lessico, sfruttano particolari dizionari detti (*lexicon*), in cui le parole sono annotate manualmente con il loro orientamento semantico. Nell'approccio Machine Learning, invece, a partire da un insieme *training* di esempi, un sistema utilizza algoritmi di intelligenza artificiale per trattare l'analisi del sentimento, come un problema di classificazione del testo.

3.1 Lexicon-based approach

Il sentimento, il più delle volte, è una funzione dell'orientamento semantico e dell'intensità delle parole usate. Le tecniche basate sul lessico funzionano sul presupposto che la polarità collettiva di una frase o di un documento sia la somma delle polarità delle singole frasi o parole.

La metodologia classica per il calcolo della somma del sentimento può essere descritta dai seguenti step:

- **Pre-Processing**. Il testo viene sottoposto a passaggi di pre-elaborazione: POS-tagging, rimozione stopword, gestione delle negazioni, tokenizzazione in N-grammi. Il risultato della pre-elaborazione è un insieme di token o Bag-of-Word.

- **Associare ogni parola alla polarità descritta nel lexicon**. Ogni parola estratta viene confrontata con il lessico. Se la parola viene trovata nel lessico, la polarità di quella parola viene aggiunta al punteggio di sentiment del testo. Se la parola non viene trovata nel lessico, la sua polarità è considerata pari a zero.

- **Calcolare lo score complessivo del testo**. Dopo aver assegnato i punteggi di polarità a tutte le parole che comprendono il testo, il punteggio finale del sentimento del testo viene calcolato dividendo la somma dei punteggi delle parole per il numero di tali parole.

Questo approccio è ulteriormente diviso in due due categorie:

- **basato su dizionari** (*Dictionary-based approach*)
- **basato su corpora** (*Corpus-based approach*)

Entrambi i metodi prevedono l'individuazione di parole semanticamente rilevanti all'interno del documento.

3.1.1 Dictionary-based approach

Il *dictionary-based* ricerca le parole all'interno di dizionari opportunamente costruiti etichettando ogni termine semanticamente rilevante con informazioni sintattico-lessicali, sinonimi e contrari ma, soprattutto, con un punteggio di orientamento del sentimento. La costruzione, solitamente manuale con l'aiuto di linguisti specializzati, parte da un insieme ridotto di *opinion word* alle quali viene assegnato un valore di polarità. Questo insieme viene allargato grazie all'integrazione di sinonimi e contrari presenti all'interno di dizionari lessicali o di appositi dizionari di sinonimi. Il dizionario viene di volta in volta arricchito con le parole che compongono il testo.

Due dei vocabolari più utilizzati:

SenticNet

Framework costituito da un insieme di strumenti e tecniche per l'analisi del sentimento che combina la psicologia, la linguistica e l'apprendimento automatico [3]. In questo contesto, SenticNet è più comunemente definito come elaborazione del *sentic*, un paradigma multidisciplinare che va oltre i semplici approcci statistici per l'analisi del sentimento concentrandosi su una rappresentazione semantica che preserva i concetti del linguaggio naturale e sulla struttura delle frasi.

SentiWordNet

Estensione del database di parole per la lingua Inglese WordNet, utilizzato per confrontare le similarità tra le parole [29], in cui assegna tre sentiment score ad ogni parola del tipo: *positivity, negativity, objectivity.*

3.1.2 Corpus-based approach

Il *corpus-based* effettua una ricerca in un corpus annotato, una raccolta solitamente molto ampia di documenti. Ogni corpus può portare alcune specificità del dominio da cui si sono raccolti i documenti.

Esistono due principali approcci di ricerca:

- **Approccio statistico**: le co-occorrenze delle parole sono calcolate per identificare il sentimento. Analizzando la frequenza in una raccolta di corpus si può identificarne la polarità: se questa appare maggiormente in testi positivi è possibile assegnarne orientamento positivo, se appare maggiormente in testi negativi è possibile assegnarne polarità negativa. Nel caso presentasse frequenze simili, la valenza solitamente associata è quella neutra.
- **Approccio semantico**: i termini sono rappresentati nello spazio semantico per scoprire la relazione tra essi [30]. L'idea di base è che le parole semanticamente "vicine", tendono ad avere anche una polarità similare.

3.2 Machine Learning approach

In questo approccio è necessario distinguere già da subito due categorie di approccio, dovute alle metodologie di apprendimento automatico:

- **apprendimento supervisionato** (*Supervised Learning*)
- **apprendimento non supervisionato** (*Unsupervised Learning*)

Nel Supervised Learning, gli esempi di allenamento forniti, sono sempre composti dalla coppia di dati contenente l'esempio originale e l'etichetta, ovvero l'output atteso. Ogni etichetta rappresenta una *classe* o categoria.
Nel Unsupervised Learning, la conoscenza è estrapolata dalle caratteristiche comuni identificate nel training set, che a differenza dell'apprendimento supervisionato, non è etichettato. Le classi quindi non sono note a priori ma vengono apprese in modo

automatico.

Le tipologie di problemi sono solitamente:

- **classificazione**: decidere a quale categoria (dato qualitativo o attributo categorico) appartiene un determinato dato. Si vuole trovare una funzione di mapping (f) da variabili di input (x) a variabili di output discrete (y).
- **clustering**: raggruppare i dati che presentano caratteristiche simili.
- **regressione**: prevedere il valore futuro (dato quantitativo o attributo numerico) di un dato. Si vuole approssimare una funzione di mapping (f) da variabili (x) a variabili di output continue (y).

3.2.1 Supervised Learning

Utilizzati principalmente per i problemi di classificazione, compito di questi algoritmi è di trovare una funzione (una regola o modello) con cui creare una relazione tra gli esempi e le etichette fornite.

Esistono differenti tipologie di classificatori:

- Classificatori lineari
- Classificatori basati su alberi di decisione
- Classificatori probabilistici
- Classificatori basati su regole
- Classificatori basati su reti neurali

Di questi, solo alcuni sono stati utilizzati e analizzati in questo lavoro di tesi. Sono stati scelti i modelli che, nella letteratura scientifica, risultano tutt'ora oggetto di studio nel campo del NLP e che hanno riportano i migliori risultati per i task della sentiment analysis.

3.2.2 Classificatori lineari

I classificatori lineari, utilizzano combinazioni lineari di vettori di features per identificare e classificare le classi di appartenenza dei dati.

Support Vector Machine (SVM)

È un classificatore lineare che fa uso del concetto di piani decisionali che definiscono i confini decisionali, dove un piano decisionale è un piano che separa un insieme di oggetti con appartenenza a classi diverse. SVM cerca un iperpiano con il maggior margine possibile. Teoricamente esistono infiniti iperpiani. Per una classificazione corretta, che permetta di separare i dati in due insiemi, è necessario determinare i parametri caratteristici $(\boldsymbol{w}, \boldsymbol{b})$ dell'iperpiano che li separa al meglio.

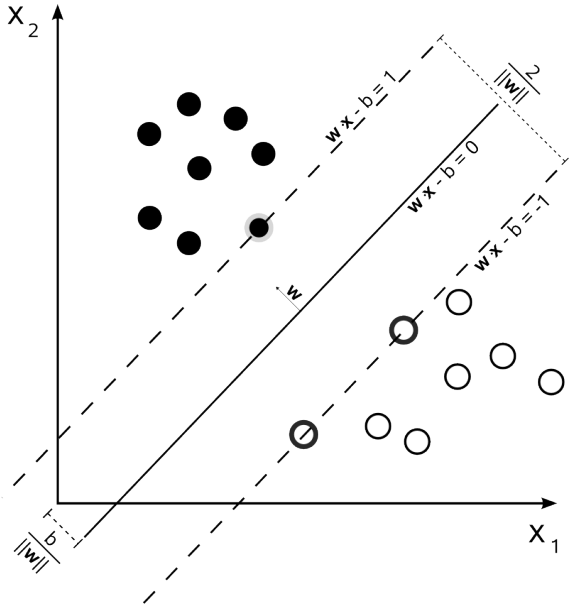


Figura 3.2: SVM

In figura 3.2 è illustrato un esempio di separazione binaria di due classi.

Dato un set di n punti nella forma $(x_1, y_1), ..., (x_n, y_n)$ dove y_i è la classe del punto x_1 di valore -1 o 1. Ogni x_1 è un vettore di valori reali. L'obiettivo è trovare l'iperpiano con margine massimo che separi i punti avente classe $y = -1$ da quelli avente classe $y = 1$. Chiamando **w** vettore *normale* all'iperpiano e b l'intercetta all'origine, i punti che giacciono sull'iperpiano separatore devono soddisfare l'equazione: $\boldsymbol{wx} - b = 0$.

Per classificare dati non linearmente separabili, è necessario ricorrere alla tecnica degli spazi immagine (*feature spaces*). Questa tecnica consiste nel mappare i dati iniziali in uno spazio di dimensione superiore.

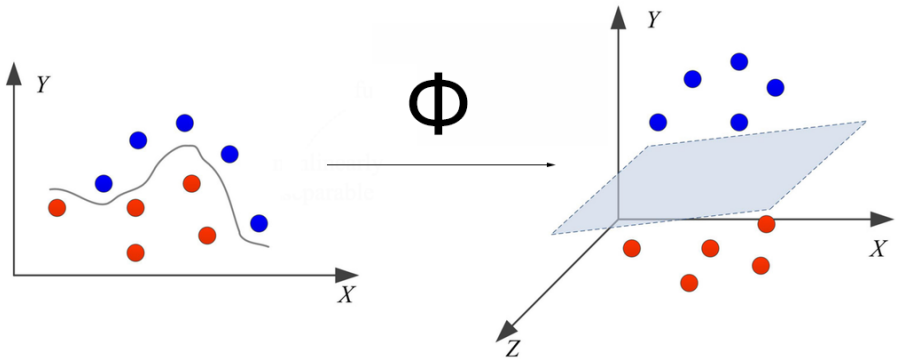


Figura 3.3: Feature space

Sia m la dimensione del nuovo spazio, con $m > n$, si mappano i punti nel nuovo spazio tramite la funzione

$$\phi = \mathbb{R}^n \rightarrow \mathbb{R}^m \tag{3.1}$$

3.2.3 Classificatori probabilistici

I classificatori probabilistici (*probabilistic classifiers*) forniscono un metodo di base di studio della classificazione statistica in domini complessi come il natural language e il visual processing. [9] I classificatori, in fase di training, memorizzano i parametri delle distribuzioni di probabilità delle classi e delle features estratte dal testo. Nella fase di classificazione, si valutano le probabilità delle feature per ogni classe.

Naïve Bayes

Naïve Bayes (NB) è un classificatore probabilistico basato sull'applicazione del teorema di Bayes. Il modello basandosi sulla distribuzione delle parole nei documenti (o come nel caso di studio, delle frasi) calcola le probabilità posteriori delle classi.

Secondo il teorima di Bayes, la probabilità che una data parola appartenga ad

una particolare classe è data dall'equazione:

$$P(label|feature) = \frac{P(label) * P(feature|label)}{P(feature)} \tag{3.2}$$

$P(label)$ è la probabilità a priori che una parola appartenga ad una classe.
$P(feature|label)$ è la probabilità che una feature sia presente in un messaggio etichettato con quella label.
$P(feature)$ è la probabilità a priori di un insieme di feature di apparire.
Il modello impara ad assegnare ad ogni parola, la probabilità di apparire una frase classificato con una certa classe (positiva, negativa, neutra). Per le frasi di test, applicherà quindi la regola sopra descritta per classificarle a seconda delle classi che ritiene essere più probabili, date le parole di cui sono composte.

$$P(label|token1, token2, token3...) = \frac{P(label) * P(token1, token2, token, ...|label)}{P(token1, token2, token3...)} \tag{3.3}$$

3.2.4 Classificatori basati su reti neurali

I classificatori basati su reti neurali utilizzano una rete neurale artificiale (o topologie di più reti neurali). Simulano le reti neurali biologiche del cervello umano prendendo ispirazione dal suo comportamento e dalle sue capacità di apprendere informazioni, prendere decisioni e classificare elementi. In seguito verranno meglio approfondite nel capitolo "Deep Learning".

3.2.5 Unsupervised Learning

Ai sistemi di Unsupervised Learning, viene chiesto di estrapolare una regola che raggruppi di esempi forniti, secondo caratteristiche che ricava automaticamente.

Non sono oggetto di studio di questa tesi ma non per questo ritenute poco importanti: una potenziale analisi di tipo non-supervisionato, può far luce su alcuni aspetti e portare gradi benefici all'apprendimento supervisionato, ad esempio, riconoscere

pattern rilevanti all'interno del testo o riconoscere come più rilevanti alcune specifiche categorie di feature.

3.2.6 Ulteriori metodologie

Tecniche combinate Vi sono alcuni approcci che utilizzano una combinazione di quelli sopra descritti.

Liu et al. (2004) [6] utilizzando due lessici di parole e dati non etichettati, ognuno relativo ad una lista di parole associate ad un sentimento (negativo e positivo), creano pseudo-documenti contenenti tutte le parole del lessico scelto. Successivamente, calcolano la somiglianza del coseno tra questi pseudo-documenti e i documenti senza etichetta. In base alla somiglianza del coseno, a un documento viene assegnato un sentimento positivo o negativo. Li usano poi per addestrare un classificatore Naive Bayes.
Un altro approccio combinato è proposta da Melville et al. (2009) [7]. Hanno usato quello che chiamano "un framework unificato" che permette di usare le informazioni lessicali di sfondo, in termini di associazioni parola-classe e perfezionare queste informazioni per ulteriori domini specifici utilizzando qualsiasi esempio di training disponibile.

Cross-Domain Sentiment Classification La classificazione è altamente sensibile al dominio da cui i training data sono estratti. Si studia come si comporta un training che utilizza un *Source Domain* per l'apprendimento (scelto solitamente perché molto grande rispetto ad altri domini disponibili) per classificare documenti di un *Target Domain*.
Diverse sono le strategie sul come mescolare i dati e su quali feature utilizzare.
Mahalakshmi et al. [8] utilizzano diversi modelli di machine learning allenati a partire da un diverso dominio di training, come ad esempio le recensioni di prodotti Amazon e le etichette associate, per predire la polarità delle recensioni TripAdvisor.

Cross-Language Sentiment Classification Lo studio mira a capire quando è possibile riutilizzare una studio di sentiment analysis effettuato a partire da una lingua su di una differente. Alcune aziende, ad esempio, vogliono comparare le opinioni dei loro prodotti e servizi in paesi differenti, senza dover costruire da capo un nuovo sistema per le nuove lingue.

Hongxia et al. [16] presentano un modello di classificazione basato su SVM allenato con esempi puramente in lingua Inglese per predire la polarità di un insieme ristretto di frasi tradotte dalla lingua Cinese.

Capitolo 4

Deep Learning

Questo capitolo fornisce una panoramica generale del *Deep Learning*, delle tipologie di reti neurali e delle tecniche utilizzate nella letteratura scientifica di maggiore successo.

Da circa un decennio a questa parte, il Deep Learning è emerso come una potente tecnica di apprendimento automatico e ha prodotto notevoli risultati in molti domini applicativi, che vanno dalla visione artificiale (*computer vision*) al riconoscimento vocale (*Speech recognition*) al Natural Language Processing (*NLP*).

Il deep learning (o apprendimento approfondito in italiano) indica un particolare approccio alla progettazione, allo sviluppo e all'applicazione delle **reti neurali artificiali** (*artificial neural networks*) a compiti di apprendimento utilizzando reti neurali a più livelli (*layer*).

4.1 Reti neurale biologica

Le reti neurali artificiali sono ispirate dalla struttura delle reti biologiche del cervello umano. Su questa base stati sviluppati modelli matematici capaci di simulare i comportamenti elementari delle reti neurali biologiche.

Una rete neurale è costituita da un grande numero di elementi computazionali, i neuroni, organizzati in livelli e fittamente connessi, che lavorano all'unisono, scam-

biandosi informazioni l'uno l'altro e in grado di variare la loro configurazione in risposta agli stimoli esterni.
Biologicamente parlando, nello specifico, all'interno di una rete neurale biologica troviamo:

- **Somi**: i corpi cellulari dei neuroni. Ricevono e processano le informazioni sotto forma di segnale elettrico da tutti i dendriti. Se la somma pesata del potenziale supera il *valore di attivazione* (o soglia) emettono a loro volta un impulso elettrico verso gli assoni a cui sono collegati in uscita;
- **Neurotrasmettitori**: composti biologici (ammine, peptidi, aminoacidi) responsabili della modulazione degli impulsi nervosi;
- **Assoni**: linea di uscita del neurone, che può essere multipla e diramarsi in migliaia di rami;
- **Dendriti**: collegamenti in ingresso del neurone;
- **Sinapsi**: punto di contatto tra due neuroni. La parte superiore trasforma il segnale elettrico in sostanza chimica (neurotrasmettitore) per essere passato dall'altra parte ed essere riconvertita in segnale elettrico. Ogni neurone ne possiede migliaia. Ogni sinapsi ha una funzione eccitatoria oppure inibitoria, facilitando o inibendo la trasmissione dell'impulso nervoso.

La configurazione della rete neurale biologica è dinamica. Significa che il numero di sinapsi può aumentare o diminuire a seconda degli stimoli ricevuti dall'esterno.

Una rete neurale artificiale è modellata come un sistema dinamico avente la topologia di un grafo orientato, in cui i nodi sono i neuroni artificiali, gli archi i collegamenti tra i nodi con associati i pesi sinaptici.

4.2 Reti neurali artificiali

Come per le tecniche viste nei capitoli precedenti, anche qua vengono distinte le due diverse metodologie caratteristiche dell'apprendimento automatico, le due regole di apprendimento: con supervisione e senza supervisione.

Apprendimento supervisionato

Alla rete viene fornito in input un training set composto di dati e le rispettive etichette delle classi (coppie input-output atteso). La rete dato un input calcola il suo output. L'errore, dato dalla differenza tra l'output calcolato e l'output atteso, serve a supervisionare l'apprendimento e modificare i pesi interni, cercando quindi di imparare e minimizzare l'errore stesso durante le epoche di allenamento.

Apprendimento non supervisionato

Alla rete vengono presentati solo valori di input e la rete li divide in gruppi usando misure di similarità, senza l'aiuto di delle etichette di output come avviene invece nell'apprendimento supervisionato.

Basandosi sulle topologie della rete, le reti neurali possono essere generalmente classificate in *feed-forward neural networks* e *recurrent/recursive neural networks*, che possono essere mixate e abbinate fra loro.

Feed-forward network

Nella rete feed-forward, gli impulsi si propagano in un'unica direzione, in avanti.
La rete feed-forward più semplice è il **percettrone a singolo strato** (*single layer perceptron - SLP*). Il modello è rappresentato in figura 4.1.
È costituito da un livello di ingresso che alimenta direttamente l'unità di output attraverso connessioni pesate e non presenta nodi nascosti.

I neuroni ricevono in input degli stimoli. I singoli input vengono moltiplicati per un opportuno valore detto *peso*. Il risultato delle moltiplicazioni viene sommato e se la somma supera una certa soglia, definita dalla funzione di attivazione, il neurone si

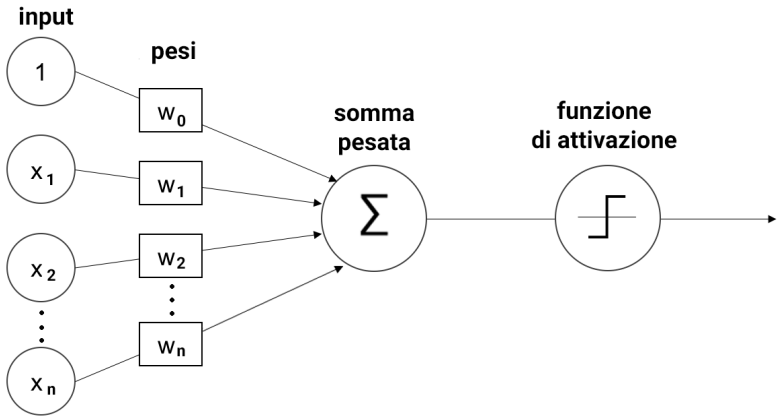


Figura 4.1: Perceptron

attiva attivando la sua uscita. Il peso indica l'efficacia sinaptica della linea di ingresso.

Una evoluzione del perceptron, è il **perceptron multistrato** (*Multilayer perceptron - MLP*). È una rete feedforward ad almeno tre livelli. Al suo interno troviamo almeno uno strato nascosto (*hidden*), dove avviene una elaborazione delle informazioni che provengono dallo strato di input, per poi essere passate al nodo ti output.

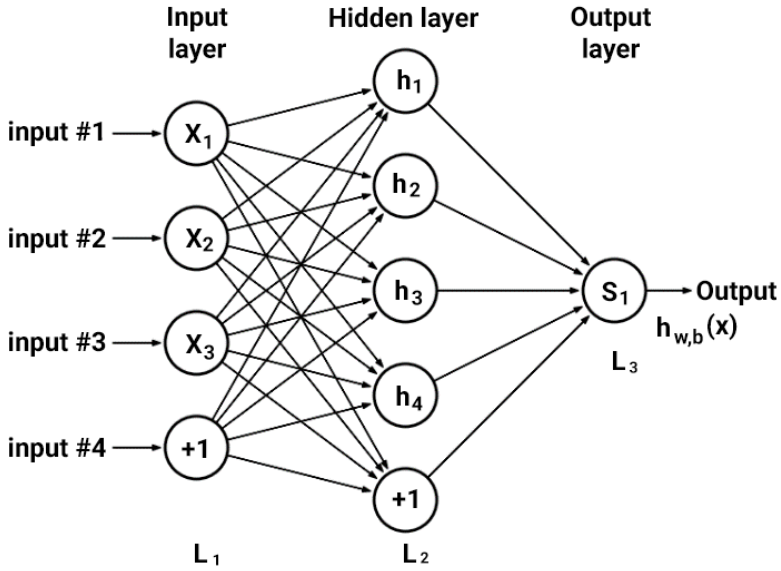


Figura 4.2: Multilayer perceptron

Ogni neurone è collegato con tutti i neuroni dello strato precedente e successivo. Nello strato nascosto non vi sono collegati fra di essi. Utilizzando la tecnica di *feedforward propagation* e *backpropagation* per l'allenamento della rete, a differenza

del SLP standard, è in grado di distinguere i dati che non sono linearmente separabili.

Un semplice esempio di rete feedforward è mostrato in figura 4.2.
La rete consiste di tre livelli, L_1, L_2 e L_3. L_1 è il livello di input, che corrisponde al vettore di input (x_1, x_2,x_3). L_3 è il livello di output, che corrisponde al vettore di putput (S_1). L_2 è il livello nascosto. Gli elementi nei livelli L_2 e L_3 rappresentano i neuroni come descritti in precedenza e aventi il ruolo di **funzione di attivazione**. Una linea tra due neuroni rappresenta una connessone per il trasporto delle informazioni. Ad ogni connessione è associato un **peso**.
L'allenamento della rete neurale è ottenuto aggiustando i pesi tra i neuroni.

4.2.1 Funzioni di attivazione

Entrando nel dettaglio dei livelli nascosti e riferendoci nuovamente alla figura 4.2, vediamo come ogni neurone in L_2 prende gli input x_1, x_2 e x_3 dal livello L_1. In output abbiamo il valore $f(W^t x) = f(\sum_{i=1}^{4} W_i x_i + b)$ con f funzione di attivazione. W_i sono i pesi delle connessioni; b è il bias.
La scelte classiche di f sono la funzione sigmoide (**sigmoid**), funzione tangente iperbolica (**tanh**) o la funzione rectified linear (**ReLU**).
La funzione sigmoide prende valori reali e comprime tali valori nell'intervallo tra 0 e 1. La tangente iperbolica ha come codominio l'intervallo da -1 a 1. La ReLU restituisce zero quando l'input è inferiore a 0.
Le equazioni delle tre funzioni sono rappresentate di seguito:

$$f(W^t x) = sigmoid(W^t x) = \frac{1}{1 + exp(-W^t x)} \tag{4.1}$$

$$f(W^t x) = tanh(W^t x) = \frac{e^{W^t x} - e^{-W^t x}}{e^{W^t x} + e^{-W^t x}} \tag{4.2}$$

$$f(W^t x) = ReLU(W^t x) = max(0, W^t x) \tag{4.3}$$

In L_3, possiamo usare la funzione **softmax** che è una generalizzazione di una funzione logistica che comprime un vettore k-dimensionale X di valori arbitrari reali in un vettore k-dimensionale $\sigma(X)$ di valori reali compresi nell'intervallo (0,1) l cui somma è 1.

$$\sigma(X)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} for j = 1, ..., k \tag{4.4}$$

4.2.2 Convolutional Neural Network

Le reti convoluzionali, Convolutional Neural Network (CNN), sono un tipo particolare di rete neurale artificiale feedforward, largamente impiegata nel campo della visione artificiale (*computer vision*). Sono ispirate alla corteccia visiva umana. La corteccia visiva contiene un grande numero di cellule responsabili della rilevazione della luce in piccole sotto regioni e sovrapposte dei campi visivi, che sono chiamati campi ricettivi. Queste celle agiscono come locali sullo spazio di input. La CNN è composta da più strati convoluzionali, ognuno dei quali svolge la funzione che viene elaborata dalle cellule nella corteccia visiva [31].

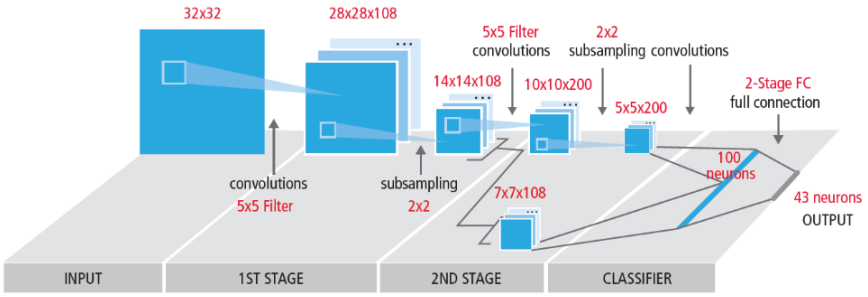


Figura 4.3: Convolutional Neural Network

Convoluzione

Il processo di **convoluzione**, divide l'immagine in vari frammenti sovrapposti. I diversi frammenti sono in seguito analizzati per individuare le particolarità che lo

caratterizzano, trasferendo l'informazione allo strato seguente sotto forma di *feature map* contenente le relazioni tra neuroni e particolarità. I livelli convoluzionali possono essere intepretati come **feature detector**, cioè strumenti che ci permettono di comprendere i dettagli delle immagini.

Pooling

Il processo di **pooling** consiste nel ridurre la dimensione spaziale delle rappresentazioni al fine di ridurre quindi il numero di parametri e di conseguenza il tempo computazionale di analisi, pur cercando di non perdere troppa precisione. Nelle immagini ad esempio, possiamo interpretare il pooling come una riduzione della qualità dei pixel: una regione di pixel si riduce ad un unico pixel. Al pixel può ad esempio venire assegnato un colore in base alla media dei colori della regione originaria.

La figura 4.3 mostra una CNN utilizzata per il riconoscimento di segnali stradali [17]. -

Le caratteristiche e gli strumenti sviluppati per le reti CNN, trovano sempre maggiore impiego, anche nel campo dell'NLP. Il punto chiave, è similarità con cui vengono estratte le features: come nelle immagini, alcuni dei punti chiave utili per identificare la classe di appartenenza, sono collocati in differenti parti dei dati di input. Ad esempio, nel task di classificazione dei documenti, una singola frase o parola, può aiutare a classificare l'argomento del documento. Siamo interessati a sapere se particolari sequenze di parole siano buoni indicatori dell'argomento, indipendentemente dalla loro posizione con cui appaiono all'interno del documento.

4.2.3 Recurrent Neural Network

Le reti ricorrenti, Recurrent Neural Network (RNN) [18], sono una classe di reti neurali in cui alcune delle connessioni di feedback dei neuroni, formano cicli diretti. I cicli della rete, permettono di analizzare sequenze temporali. Rieseguire lo stesso compito, permette di "ricordare" informazioni su ciò che è stato elaborato in precedenza, consolidando i pesi in relazione alle caratteristiche che risultano essere più o meno importanti per la classificazione finale.

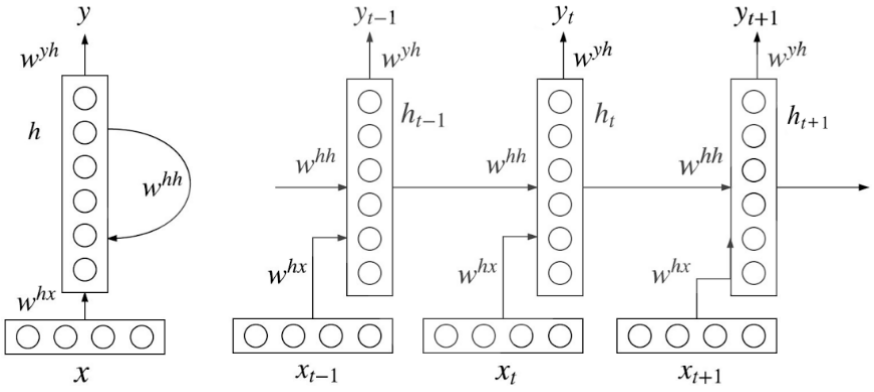


Figura 4.4: Recurrent Neural Network

La Figura 4.4 mostra un esempio di un RNN. La rete a sinistra è una rete dispiegata (*unfolded*) con cicli, mentre la rete di destra è una rete di sequenza *folded* con tre fasi temporali. La lunghezza delle fasi temporali è determinata dalla lunghezza dell'input. Ad esempio, se la sequenza di parole da elaborare è una frase di sei parole, l'RNN verrebbe dispiegata in una rete neurale con sei livelli. Ogni livello corrisponde ad una parola.

A differenza di una rete neurale feedforward, che utilizza parametri diversi per ogni livello, RNN condivide gli stessi parametri in tutti i passaggi. Ciò significa che esegue lo stesso compito in ogni fase, solamente con input diversi. Ciò riduce notevolmente il numero totale di parametri necessari per l'apprendimento.

4.2.4 LSTM

La Long Short Term Memory (LSTM) [19], è una particolare rete RNN proposta da Hochreiter e Schmid- huber, che è capace di imparare le dipendenze a temporali lungo termine (*long-term dependencies*). A differenza delle RNN classiche, la catena di ripetizioni dei livelli è più complicata. In figura 4.5 mostra l'architettura della rete LSTM. L'architettura più comune, è formata da unità di memoria (*memory cell*), che permettono il salvataggio delle dipendenze temporali a lungo termine.

Ogni blocco è formato da quattro unità: tre layer detti *gate* e una cella di memoria con una connessione con se stessa. I gate servono a modulare le interazioni tra la cella di memoria e l'esterno del blocco. Determinano se un input debba essere salvato

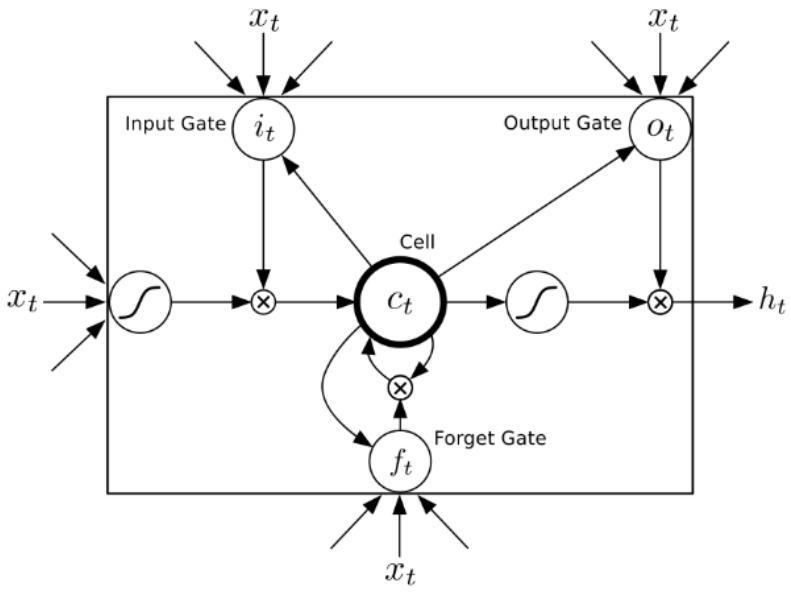


Figura 4.5: Architettura LSTM [20]

nella cella di memoria, per quanto tempo deve essere mantenuto e quando deve essere emesso. Il gate di input (*input gate*) interagisce con l'esterno permettendo l'ingresso al segnale di input. Il *forget gate* determina quando ricordare o dimenticare lo stato della cella. L'*outputgate* consente l'uscita dello stato di memoria della cella, fornendolo ad altri blocchi.

4.2.5 Modelli Concatenati

La potenza delle deep neural networks non deriva soltanto dalla loro profondità ma anche dalla flessibilità nel comporre strutture complesse. Oltre ai livelli e alle topologie già descritte per reti che sfruttano una tipologia di batch di input, è possibile gestire features di input diverse. Un esempio sono le *reti siamesi*, una classe di architetture di rete neurali che contengono due o più sottoreti identiche, ovvero hanno la stessa configurazione con gli stessi parametri e pesi. L'aggiornamento dei parametri è speculare su entrambe le sottoreti. Le reti siamesi sono popolari per le attività che comportano la ricerca di somiglianza o di relazione tra oggetti comparabili. Similmente alle reti siamesi, è possibile creare architetture di reti parallele (o **modelli concatenati**), ma che a differenza loro, non condividono i pesi ma prevedono un livello di concatenazione. I modelli concatenati permettono di gestire

tipologie di features diverse, adottando per ognuna di esse una rete appositamente implementata per estrarne quanti più dettagli utili alla classificazione.

La libreria Keras, successivamente presentata, prevede la funzione **Concatenate()** per poter concatenare output provenienti da reti diverse.

4.3 Allenamento

Il modello, per allenarsi, lavora unicamente sul training set. Se il modello, anziché allenarsi apprendendo la miglior supervisione, insegue l'andamento dei dati di training "memorizzandoli", c'è il rischio che in futuro non generalizzi. Questa condizione viene chiamata **overfitting**.

La loss function permette di introdurre termini di **regolarizzazione** per penalizzare alcuni errori e prevenire l'overfitting.
La loss function viene utilizzata in fase di training per valutare la bontà delle previsioni del modello. Misura l'errore tra i valori predetti e i valori reali. Deve essere definita in base al tipo di problema.

4.3.1 Ottimizzazione

Definito il modello di rete neurale e la loss function, si procede alla fase vera e propria di allenamento della rete. Per effettuare la miglior classificazione, la rete modifica i pesi in modo tale da minimizzare la loss.
Ci chiediamo quando fermare l'addestramento della rete. Minimizzare l'errore sul training set non equivale ad ottenere una migliore capacità di generalizzazione. È necessario fermare l'aggiornamento dei pesi prima che il modello porti ad avere overfitting.

È possibile guidare il processo di ricerca verso la direzione di massima discesa della funzione di loss. Questo approccio è il metodo di **discesa di gradiente** (*gradient descent*).

Gli ottimizzatori sono algoritmi di aggiornamento dei pesi delle reti neurali nelle fasi di training. Il **Gradient Descent** è uno degli algoritmi più popolari per l'ottimizzazione. Esistono differenti librerie in Keras che forniscono diverse implementazioni per l'ottimizzazione del gradient descent. Di seguito sono riassunti gli algoritmi utilizzati:

- **SGD** (Stochastic gradient descent)
  Mantiene un singola learning rate (chiamato alfa) per tutti gli aggiornamenti di peso durante l'allenamento.

- **Adagrad** (Adaptive Gradient Algorithm)
  Mantiene un learning rate per parametro che migliora le prestazioni in caso di problemi con gradienti sparsi.

- **RMSProp** (Root Mean Square Propagation)
  Mantiene fissi i learning rate per parametro adattati in base alla media della variazione dei gradienti.

- **Adam** (Adaptive moment estimation)
  Invece di adattare i learning rate alla media come in RMSProp, usa anche la varianza non centrata.

- **AdaDelta**
  Estensione di Adagrad che cerca di ridurre in maniera monotona il learning rate.

4.4 Keras

Per l'implementazione delle reti neurali è stata scelta la libreria Python *Keras*. Keras si basa sul framework *Tensorflow* e mette a disposizione numerose funzioni per la creazione delle reti neurali, per l'allenamento e il salvataggio dei pesi.

Keras Callbacks

Keras fornisce un insieme di funzioni di *Callback* da applicare alle procedure di training. Le callback permettono di osservare e raccogliere statistiche interne, determinare parametri di stop dell'apprendimento, salvare lo stato dell'allenamento. È possibile passare un elenco di callback al metodo *.fit()* al momento della inizializzazione dela fase di training.
Le callback utilizzate per le reti neurali implementate sono:

- **EarlyStopping**
  Arresta il training dopo un numero di epoche consecutive specificate nel parametro *patience* in cui non vi è stato un miglioramento della metrica specificata dal parametro *monitor*. Ad esempio:
  EarlyStopping(patience=5, monitor="val_loss")
  il training si arresterà se dopo 5 epoche consecutive, il training non riporta un calo della loss sul validation set.

- **ModelCheckpoint**
  Salva il modello dopo ogni epoca o mantiene semplicemente il miglior modello trovato fino all'istante di allenamento corrente, in base alla metrica definita dal parametro *monitor*.

- **CSVLogger**
  Salva le statistiche delle metriche di ogni epoca di allenamento su di un file csv.

Capitolo 5

Implementazione

In questo capitolo vengono descritti i dataset utilizzati, gli obiettivi implementativi, l'ambiente di sviluppo, i modelli di apprendimento automatico e infine le metriche di valutazione adottate nella fase di valutazione dei risultati.

Il progetto ha come obiettivo la costruzione di un sistema automatico per la classificazione della polarità di testi a livello di frase. Per questo task, sono stati scelti e analizzati due diversi domini delle frasi. Il primo, il più corposo, riguarda la recensioni di film rilasciate online. Il secondo dominio riguarda i tweet. Per ognuno di questi domini è stato analizzato lo stato dell'arte presente in letteratura scientifica, le tecniche di pre-processing e di feature extraction a loro dedicate e le tecniche di classificazione che hanno riportato i risultati migliori e più interessanti, comparate secondo le diverse metriche di valutazione tipiche degli studi di apprendimento automatico.

5.1 Ambiente di sviluppo

Il linguaggio di programmazione scelto è **Python**.

Le principali librerie utilizzate sono:

- **Keras 2.1.6** sopra l'ambiente **Tensorflow 1.8.0**

- **NumPy 1.15**
- **Scikit-Learn 0.19.2**
- **NLTK 3.3**
- **Gensim 3.5.0**
- **Pandas 0.22.0**

5.2 Internet Movie Database (IMDb)

Il primo dominio analizzato per la sentiment analysis, riguarda le recensioni di film rilasciate online nel sito imdb.com e raccolte nel dataset IMDb da Maas et al. [10].

Il dataset è composto di 50.000 recensioni, catalogate come positive o negative.

Per la catalogazione, è stato adottato un metodo automatico che guardasse unicamente a rating esplicito delle recensioni, espresso in numero di stelle. Le recensioni con un numero di stelle superiore a sei, sono catalogate come positive, le recensioni con un numero di stelle inferiore a cinque come negative.

Il dataset è disponibile nella libreria Keras, dove ogni frase è codificata nel formato *one-hot-encoding*: ad ogni parola è associato un numero univoco, utile ad esempio per l'embedding sviluppato nelle reti neurali, il quale richiede codifiche di soli interi. Per poter analizzare al meglio la vera natura delle frasi e non perdere la semantica, è stato ripristinato il dataset alla sua forma originale, così da ottenere le frasi originali composte dal solo testo.

Le frasi sono unicamente in lingua inglese; il testo è in forma minuscola. Nessun film ha più di 30 recensioni a lui dedicate.

I campi del dataset sono:

- *id*: ID univoco per ogni recensione;
- *sentiment*: polarità della recensione. 1 per le recensioni positive e 0 per quelle negative;
- *review*: testo della recensione.

Alcune righe del dataset:

id	sentiment	text
1589	1	this film was just brilliant casting location scenery story direction everyone's really suited the part they played and you could just imagine being there robert redford's is an amazing actor...
2356	0	this so called remake is terrible i went to see this tonight on the first day as the anticipation and hype was too much for me to handle but within the first half an hour we knew that this is a disaster it not only does not match the hype created but also insults the original blockbuster the script had loopholes the editing was untidy quite a few times mohanlal who is an excellent actor did an okay job...
7200	0	i saw this film awhile back while working on a trailer for the film's production company and it was terrible hewitt is mediocre at best hopkins phones his performance in but still blows away hewitt in their scenes together and alec looks bored trust me on this you should avoid this film like the plague if it ever gets released it seems to go on forever as the tired plot...

5.3 Twitter SemEval 2017

Twitter

I messaggi, chiamati *tweet*, hanno una lunghezza massima di 140 caratteri oltre a un possibile campo per le immagini. Possono contenere oltre al semplice testo, riferimenti a contenuti multimediali, riferimenti ad atri utenti o topic. I riferimenti ad altri utenti, chiamati *tags* o *mentions*, sono possibili grazie al simbolo "@" anteposto

al nickname dell'utente menzionato. Per indicare parole chiave o argomenti (*topic*) rispondendo o creando nuovi dibattiti, viene utilizzato il simbolo "#" chiamato *hashtag*. Gli hashtag permettono agli utenti di seguire gli argomenti a cui sono interessati: ricercando una keyword, è possibile ritrovare i tweet che utilizzato tale parola preceduta dal #.

Twitter dataset

A differenza delle recensioni dei film, in cui è automatico assegnare una classe di polarità perchè esplicita dal rating espresso in numero di stelle, per i tweet la classificazione è più complicata. Non vi è un rating esplicito che possa raggruppare seppur grossolanamente, i tweet nelle tre categorie di polarità positiva/negativa/neutra.

Uno dei dataset più utilizzati legato ai tweet, è *Sentiment140*, una raccolta di 1.8 milioni di frasi classificate per polarità. La sua criticità risiede nel metodo automatico di classificazione: banalmente, è stata assegnata la classe positiva ai tweet contenenti un numero di emoji "positive/sorridenti" superiore al numero di emoji restanti, viceversa per le emoji "negative" o per i tweet che ne presentano un numero simile o esenti da emoji catalogati quindi come neutri. Questa tecnica, seppur veloce, risulta essere grossolana per quanto riguarda la vera natura della sentiment analysis. Per questo lavoro di testi, si sono ricercati dataset, che seppur di ridotte dimensioni, presentassero una catalogazione di tipo manuale.
Esistono diverse competizioni che hanno lo scopo di testate i sistemi di classifcazione e raccogliere gruppi di ricerca provenienti da tutto il mondo. Annualmente, per il task **SemEval**, vengono organizzate campagne di valutazione e competizioni tra sistemi di analisi semantica, spesso sponsorizzate da compagnie private. I dataset forniti per la fase di training, sono catalogati manualmente.

Il dataset scelto è relativo alla campagna SemEval-2017 e contiene 4000 diversi tweet, catalogati per polarità.

Tutti i tweet sono in lingua inglese.

I campi del dataset sono:

- *id*: ID univoco per ogni tweet;
- *sentiment*: polarità della frase. 1 per i tweet positivi e 0 per quelli negativi;
- *text*: testo del tweet.

Alcune righe del dataset:

id	sentiment	text
23366	1	one ticket left for the AT_USER game tomorrow! don't miss the rematch of the nfc championship game against the ny giants! hit me up!
23366	0	noel gallagher: "musicians are fucking idiots." - yeah, brian may - he's an idiot... an idiot who just so happens to be a professor?
23366	1	just watched the campaign with will ferrell and zack galafianakis... good fun for a wednesday. hilarious

Pre-processing

- Le menzioni effettuate utilizzando il simbolo @ sono rimpiazzate dal token "AT_USER;
- Gli url sono rimpiazzati dal token URL;
- I caratteri "#" e "@" sono stati rimossi.

5.4 Tokenizzazione

La tokenizzazione (*tokenization* in inglese), come già accennato nei capitoli precedenti, è il processo di suddivisione delle frasi nei token che le compongono. Tokenizer a livelli di granularità superiore, possono ottenere non soltanto gruppi di token ma gruppi di di frasi che compongono i documenti. Nel NLP è considerata una delle fasi più cruciali, perché non è sempre possibile ottenere suddivisioni univoche, ma se

ne possono ottenere diverse a seconda degli algoritmi utilizzati, spesso studiati per diversi domini applicativi.

Si sono sperimentati diversi tokenizer. ad esempio, per la frase
"I've had better weekends, but whatever, I'm going to use today to sleep and watch dexter"

- **Whitespace tokenizer**
  Suddivide il testo semplicemente osservando gli spazi, tabulazioni o nuove righe.
  ("I've", 'had', 'better', 'weekends.', 'but', 'whatever,', "I'm", 'going', 'to', 'use', 'today', 'to', 'sleep', 'and', 'watch', '#dexter')

- **Nltk TweetTokenizer**
  Tokenizer studiato appositamente per il dominio Twitter.
  ("I've", 'had', 'better', 'weekends', '.', 'but', 'whatever', ',', "I'm", 'going', 'to', 'use', 'today', 'to', 'sleep', 'and', 'watch', '#dexter')

- **Nltk RegexpTokenizer**
  Tokenizer che permette di specificare una regular expression.
  ad esempio: `[RegexpTokenizer(@"\w*\'*\w\$[\d\.]+\S+")]`
  ("I'v", 'e', 'had', 'better', 'weekends', '.', 'but', 'whatever', ',', "I'm", 'going', 'to', 'use', 'today', 'to', 'sleep', 'and', 'watch', '#dexter')

- **Nltk TreebankTokenizer**
  Tokenizer che utilizza particolari regular expressions studiate e adottate per i Treebank projects [11].
  'I', "'ve", 'had', 'better', 'weekends.', 'but', 'whatever', ',', 'I', "'m", 'going', 'to', 'use', 'today', 'to', 'sleep', 'and', 'watch', '#', 'dexter'

Come si può osservare, le tokenizzazioni non sono univoche. Risulta inoltre difficile studiare regular expressions che possano essere adottate per una tokenizzazione generale per ognuno dei dataset. Tra quelli sopra citati, è stato scelto, per entrambi i dataset, TreebankTokenizer fornito dalla libreria Nltk.

5.5 Definizione vocabolario

Al fine di ottenere le feature per i modelli di Bag-of-word, Tf-Idf e Embedding, è importante definire un vocabolario dei token. La costruzione del vocabolario consiste nel elencare univocamente tutti i token presenti nel dataset. Per migliorare l'efficienza e il tuning dei parametri, la costruzione è stata effettuata una sola volta. L'elenco presenta, oltre ai token, la loro frequenza riscontrata nel dataset e un indice univoco. I token si presentano in ordine di frequenza per facilitare la riduzione della dimensionalità.

Parte del vocabolario estratto dal IMDb dataset:
0 the 24791
1 film 13685
2 they 10316
3 good 9405
...
11 bad 5778
12 acting 5346
13 these 4008
...
26 funny 3228
27 action 2223
28 worst 2214
...

Come si può vedere, sono presenti anche le stopword come ad esempio "the" e "they".

Data una lista di sentences il dizionario è stato creato con la libreria Gensim:

Listing 5.1: Creazione dizionario

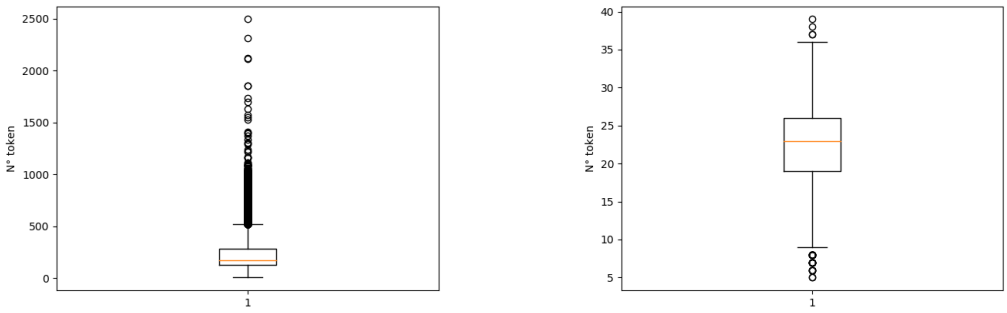
```
from gensim import corpora
```

```
m_dict = corpora.Dictionary(sentences)
```

5.6 Statistiche

Prima di procedere con l'estrazione delle feature, si sono volute studiare alcune statistiche per ognuno dei dataset.

Numero di token per frase



(a) IMDb **(b)** SemEval

Figura 5.1: Numero medio di token per frase

a) IMDb

Media token: 234.76

Deviazione standard: 172.92

b) SemEval

Media token: 22.33

Deviazione standard: 5.58

Partizione classi

a) IMDb

Recensioni positive: 25000

recensioni negative: 25000

b) SemEval

Tweet positivi: 2000

Tweet negativi: 2000

5.7 Rappresentazione delle features

Come già menzionato, i classificatori richiedono che i dati siano espressi tramite vettori di feature.

Le principali features estratte e sul quale si sono fatte le principali prove di tuning, ad esempio relative alla dimensione del vocabolario sono:

Per le seguenti rappresentazioni, viene utilizzata come esempio la frase:
"John likes to watch movies. Mary likes movies too."

- **BOW**

  Per ogni frase viene creato un vettore lungo quanto il dizionario o una sua sottoparte. Una possibile sottoparte ha grandezza **n** e riguarda i primi **n** token più frequenti. Tale vettore ha valori interi. I due possibili metodi di creare una rappresentazione del tipo Bag-of-word riguardano i valori che tali interi possono assumere:

  Dizionario di esempio: *["film","likes","movies",...,"bad","actor"]*

  a) **Frequenza**: la *i-esima* unità del vettore ha valore pari alla frequenza del token *i-esimo* all'interno della frase.

  [0,2,2,...,0,0]

  b) **Presenza binaria**: la *i-esima* unità del vettore ha valore pari a 1 se il token *i-esimo* compare all'interno della frase, 0 altrimenti.

[0,1,1,...,0,0]

- **Tf-Idf**

  la *i-esima* unità del vettore ha valore pari alla *term frequency-inverse document frequency* calcolata.

  [0,1.2,0.4,...,0,0]

- **Word embedding**

  La tecnica scelta per l'embedding delle parole è quella proposta da Keras con il suo embedding layer. Tale livello richiede che i vettori di input siano codificati come vettori di interi (tutti della stessa lunghezza) ed ogni parola sia rappresentata da un intero univoco.
  Keras fornisce la funzione *one_hot()* per codificare ogni parola.
  Dato un vocabolario di dimensione n e definito uno spazio vettoriale di embedding pari a e, ogni frase viene rappresentata da un vettore lungo e e i valori che assume sono compresi tra 0 ed n. Per le parole non presenti nel dizionario, le celle restanti vengono inizializzate al valore 0. Tale tecnica si chiama *padding* ed è fornita anch'essa da keras con la funzione *pad_sequences()*.

5.8 Riduzione della dimensionalità

Se si utilizzassero tutti i termini del vocabolario per le rappresentazioni delle feature, i vettori, oltre ad avere dimensioni enormi, risulterebbero essere certamente sparsi. Ad esempio, nella rappresentazione bag-of-word, se si utilizzassero vettori di dimensione pari al vocabolario, con corpus grandi, questi potrebbero raggiungere la dimensione del vocabolario della lingua delle frasi in esame [25]. Diminuire la cardinalità delle feature può diminuire drasticamente lo spazio in memoria e il tempo sia di estrazione delle feature stesse che di elaborazione da parte degli algoritmi di classificazione. Il

vantaggio ancor più importante nell'eliminazione delle feature ridondanti o portatrici di scarso significato, è quello di permettere al classificatore di generalizzare più facilmente e ridurre l'overfitting.

Frequenza dei token

La scelte più frequente e che si trova quasi sempre al primo posto nella pipeline della selezione delle feature, è la selezione dei token aventi la frequenza di apparizione (nel corpus di training) al di sopra di una determinata soglia. Prendendo spunto dalle legge di Zipf [13], non solo si possono omettere i termini poco frequenti, ma anche quelli nelle prime posizioni. La selezione delle soglie di upper-bound e lower-bound è stata effettuata per la creazione delle feature ed analizzata nella sezione di valutazione.

Stopwords

Nella categoria dei termini troppo frequenti, ricadono quei termini che come è già stato citato, vengono chiamati *stopword*. La creazione del vocabolario con Gensim permette di omettere fin da subito una lista di stopword. La scelta implementativa è stata di non omettere queste parole nel vocabolario iniziale, ma permetterne l'eventuale rimozione solo in fase di costruzione dei vettori di feature. Una prematura rimozione, non avrebbe permesso di mantere la semantica utile alla tecnica di word embedding o di costruire ad esempio bi-grammi o tri-grammi come "**not**-very" o "was-**the**-best".

Stemming

Lo stemming, la riduzione alla radice morfologica delle parole, seppur spesso utile nel ridurre il numero di termini, non è stata applicata per lo stesso motivo per cui non sono state tolte le stopword.

5.9 Arricchimento features

5.9.1 POS-tagging

Per arricchire le frasi dei pos-tag, è stata utilizzata la libreria Nltk. Tokenizzate le frasi e aggiunti i pos-tag ad ogni token, sono state copie dei dataset contententi le sequence dei token nel formato *"token_POS-Tag"*. Così come per il dizionario classico, è stato creato un dizionario dei token arricchiti, per facilitare la creazione dei vettori di feature nel formato BOW.

5.9.2 Features di alto livello

Oltre alle feature legate al linguaggio naturale più comunemente utilizzate e documentate documentate nella letteratura scientifica per la classificazione della polarità, possono nascere numerose altre feature che astrattamente ricadono in classi di più alto livello.

Alcune delle feature estratte e utilizzate:

- Numero token per frase
- Punteggiatura
  - Numero punti escalamativi
  - Numero punti di domanda
  - Numero punti e virgole
- Numero (o percentuale) lettere maiuscole
- Numero emoji

  Uno studio più approfondito sulle emoji, specialmente in domini legati ai social network, dove le emoji in formato unicode o *manoscritte* (ad esempio ":-)") assumono un ruolo importante tanto quanto le parole, potrebbe esplorare livelli di feature interessanti per la polarità delle frasi, come:
  - BOW emoji

- Numero emoji
- Numero emoji per macrocategoria (emoji positive/negative/neutre)

Data la natura dei dataset, in cui il numero di emoji è davvero esiguo, non è stato sperimentato tale livello di feature.

5.10 Modelli di Machine Learning

Terminato il processo di estrazione delle feature, per la classificazione della polarità, sono stati definiti diversi modelli di apprendimento automatico. Per i risultati si rimanda al capitolo successivo.

5.10.1 Naive Bayes

Il modello Naive Bayes è fornito dalla libreria Sklearn. Il modello scelto è **BernoulliNB**.

5.10.2 SVM

Anche in questo caso, i modelli sono forniti da Sklearn: **SVC** e **LinearSVC**.

5.10.3 Reti neurali

Utilizzando la libreria Keras, sono state implementate differenti reti neurali. Si differenziano prima di tutto per tipologia, dal semplice multiperceptron alla rete CNN. Per ogni tipologia si sono sperimentate diverse topologie aggiungendo più livelli nascosti, variando il numero di neuroni, utilizzando differenti funzioni di attivazione e differenti livelli aggiuntivi come il pooling o la normalizzazione.Di seguito sono riportate le tre topologie che hanno ottenuto i risultati migliori.

Topologie

a) MLP

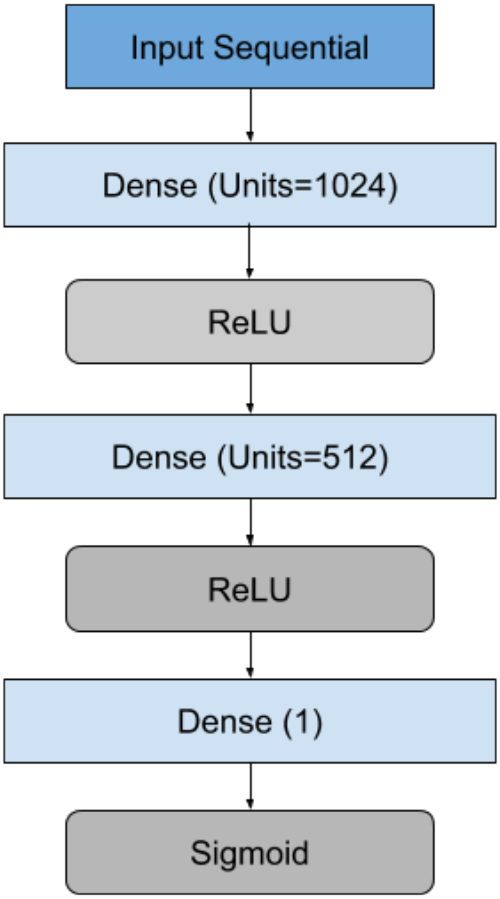


Figura 5.2: MLP architecture

b) CNN Embedding

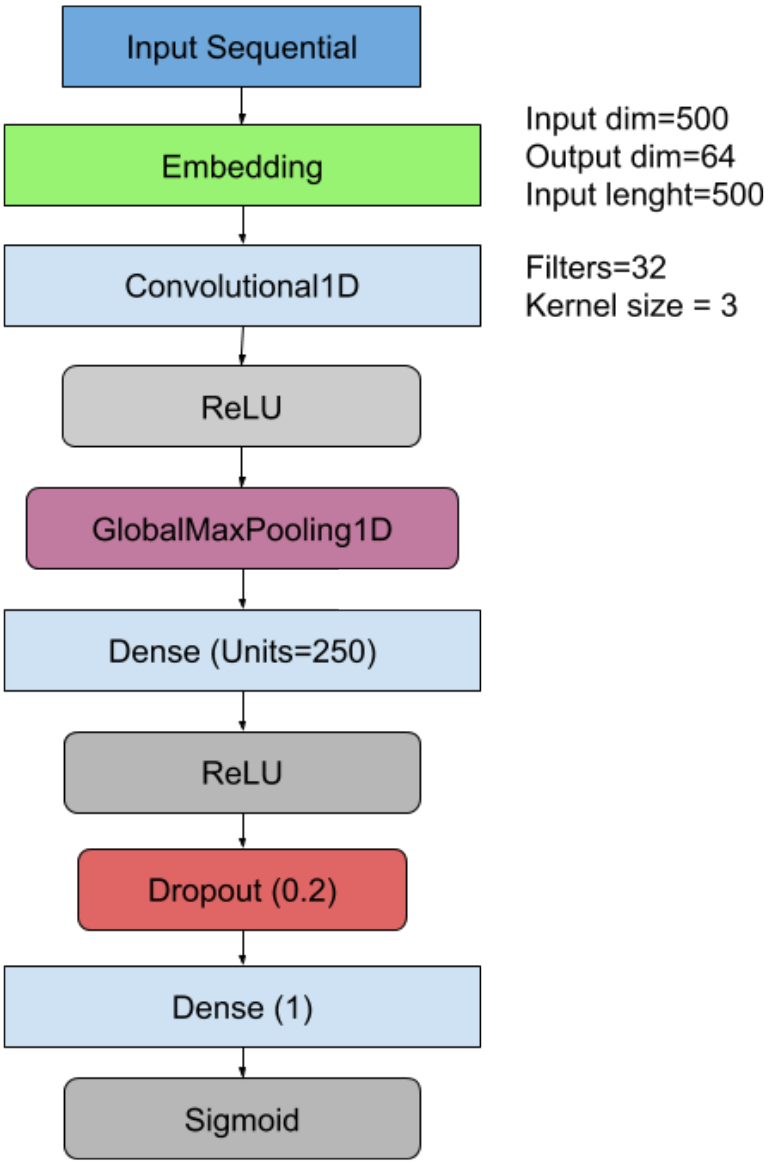


Figura 5.3: CNN architecture

c) Modelli Concatenati

Per ogni rappresentazione delle features, viene utilizzata la rete che presa tale rappresentazione singolarmente, ha ottenuto l'accuracy sul test set migliore.

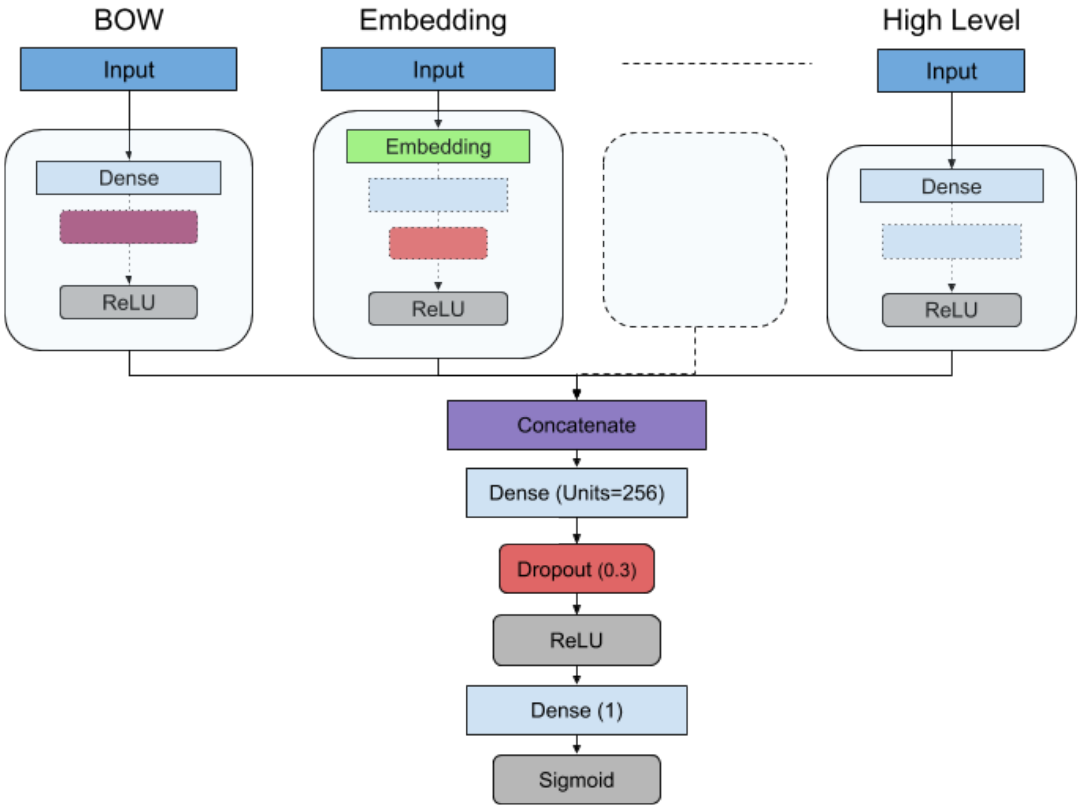


Figura 5.4: Concatenate architecture

5.10.4 Pipeline tuning iper-parametri

Per i modelli forniti dalla libreria Sklearn, è stata utilizzata la funzione *GridSearchCV* per sperimentare un tuning dei parametri che, definiti i parametri dei modelli e i loro range di valori, fornisce un metodo automatico di esecuzione dei modelli.
Ad esempio, per il modello SVC, definiti i parametri e la lista dei valori che assumeranno:

- C: [0.001, 0.01, 0.1, 1]
- Kernel: ["linear", "rbf"]
- Gamma: [1e-3, 1e-4]

GridSearchCV esegue il modello per ogni combinazione possibile di valore e fornisce i migliori settaggi che hanno contribuito ad ottenere le migliori precision e recall.

5.10.5 Cross validation

Per entrambi i domini, la partizione dei dati di **training** e quelli di **test** è del **50%**. Questa scelta è guidata principalmente dal dominio IMDb [14]: questa partizione è utilizzata tipicamente come standard per le prove di benchmark nella letteratura scientifica per le prove che riguardano questo preciso dataset di recensioni di film.

Per la fase di training, la percentuale di dati per l'**evaluation** set (estratti dal training set) è del **33%**.

5.10.6 Pre-trained GloVe Word Embedding

Come già menzionato nella descrizione della tecnica di word embedding, è possibile utilizzare livelli di embedding precedentemente allenati. GloVe [32] fornisce vettori di embedding, allenati grazie ad un dataset composta da più di 6 miliardi di token. Il dizionario utilizzato presenta 400 mila token unici. I vettori di embedding sono disponibili in quattro dimensioni: 50, 100, 200 e 300.

5.11 Metriche

Per la valutazione di un classificatore, è fondamentale definire alcune metriche. Si vuole capire se il sistema rappresenta una buona scelta, cioè si vuole misurare le sue capacità di classificare correttamente i dati, nel nostro caso, la capacità di assegnare la polarità corretta alle recensioni dei film o ai tweet chiamata *capacità di generalizzazione*. La capacità di generalizzazione è la capacità di classificare correttamente anche dati mai visti prima (non presenti nel training set). Se il sistema ha imparato i pattern del training set così da non generalizzare, si parla di *overfitting*.

Gli errori di classificazioni possono essere calcolati con la matrice di confusione. Un esempio è la figura 5.5, in cui vi sono i raggruppamenti dei dati classificati.

- **TP (true positive)**: numero di frasi positive classificate correttamente come positive;

- **TN (true negative)**: numero di frasi negative classificate correttamente come negative;
- **FP (false positive)**: numero di frasi negative classificate erronamente come positive;
- **FN (false negative)**: numero di frasi positive classificate erroneamente come negative.

Accuratezza

L'accuratezza (*Accuracy*) calcola la percentuale di frasi classificate correttamente.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{5.1}$$

Precisione

La *precision* misura la percentuale di etichette classificate positivamente rispetto al totale delle frasi etichettate in tale classe.

$$precision = \frac{TP}{TP + FP} \tag{5.2}$$

Recall

La recall è il rapporto tra quante etichette sono state classificate positivamente rispetto a quanti esempi realmente positivi sono stati forniti.

$$recall = \frac{TP}{TP + FN} \tag{5.3}$$

F-score

F-score è la media geometrica tra precision e recall.

$$F = \frac{2 * precision * recall}{precision + recall} \tag{5.4}$$

Receiver Operating Characteristic (ROC)

Le curve Receiver Operating Characteristic (ROC), sono degli schemi grafici per un classificatore binario che ne illusrano capacità diagnostica. Lungo i due assi si possono rappresentare la sensibilità e (1-specificità), come *True Positive Rate* (TPR, frazione di veri positivi) e *False Positive Rate* (FPR, frazione di falsi positivi). Si studiano i rapporti fra allarmi veri (hit rate) e falsi allarmi. Esempio classico è lo studio di un insieme di pazienti e si vuole predirre il numero dei pazienti malati e quelli sani.

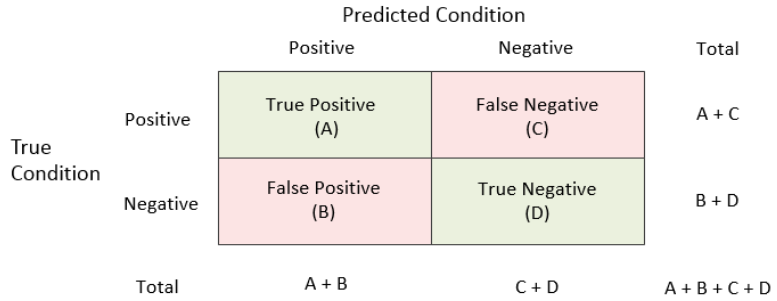


Figura 5.5: Tabella di contingenza

Si può rappresentare questo tipo di situazione utilizzando una tabella di contingenza come in figura 5.5, dove le colonne rappresentano la distinzione tra soggetti sani e malati; le righe invece rappresentano il risultato del test sui pazienti.

Area Under the Curve (AUC)

Attraverso l'analisi delle curve ROC si valuta la capacità del classificatore di discernere, tra un insieme di popolazione sana e malata, calcolando l'area sottesa alla curva ROC (AUC). AUC ha un valore compreso tra 0 e 1 ed equivale alla probabilità che il risultato del classificatore applicato ad un individuo estratto a caso dal gruppo dei malati sia superiore a quello ottenuto applicandolo ad un individuo estratto a caso dal gruppo dei sani.[12]

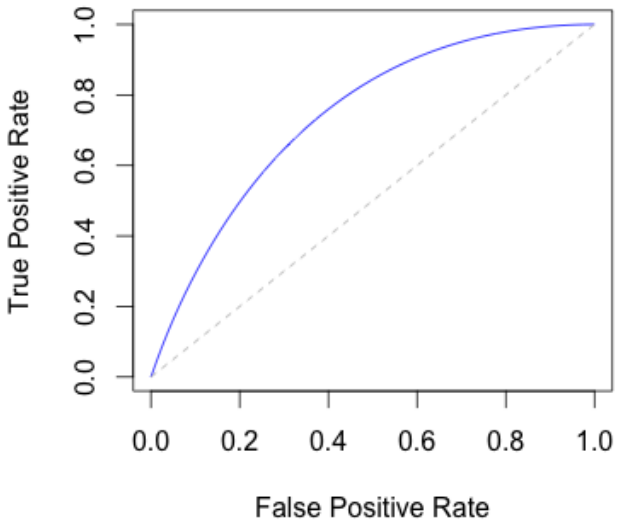


Figura 5.6: ROC

Capitolo 6

Risultati

Qui sono riportati i risultati sperimentali dei diversi modelli per la classificazione della polarità delle frasi, divisi per dataset.

6.1 IMDb

BernoulliNB

Features stack	Accuracy	Precision	Recall	F-score	AUC
BOW	0.775	0.78	0.78	0.78	0.858
Tf-Idf	0.786	0.79	0.79	0.79	0.868
w2v	**0.781**	**0.78**	**0.78**	**0.79**	**0.875**
POS	0.773	0.78	0.77	0.77	0.857
BOW + Tf-Idf	0.779	0.78	0.78	0.78	0.863
BOW + w2v	0.723	0.73	0.72	0.72	0.805
BOW + HL	0.773	0.77	0.77	0.77	0.855
BOW + Tf-Idf + w2v + POS	0.765	0.77	0.77	0.77	0.847

Tabella 6.1: Risultati BernoulliNB - Imdb Dataset

SVM

Iper-parametri migliori ottenuti tramite GridSearchCV:

- C: 0.1
- kernel: lineare

Features stack	Accuracy	Precision	Recall	F-score	AUC
BOW	0.833	0.83	0.83	0.83	0.913
Tf-Idf	0.841	0.84	0.83	0.84	0.903
w2v	0.840	0.83	0.83	0.83	0.900
POS	**0.845**	**0.84**	**0.84**	**0.84**	**0.921**
BOW + Tf-Idf	0.835	0.84	0.84	0.84	0.913
BOW + w2v	0.736	0.74	0.74	0.74	0.814
BOW + HL	0.835	0.83	0.83	0.83	0.915
BOW + Tf-Idf + w2v + POS	0.794	0.79	0.79	0.79	0.874

Tabella 6.2: Risultati SVM - Imdb Dataset

Dimensione vocabolario

Variando la dimensione del vocabolario (il numero di token unici da prendere in cosiderazione, tra i primi in ordine di frequenza tra tutti i token estratti dal training set), la miglior accuracy media sul validation set utilizzando la rappresentazione BOW, come mostrato in figura 6.3, è stata ottenuta con una dimensione pari a 500. Tale dimensione è risultata essere in media la migliore anche nelle prove a seguire.

6.1.1 Reti neurali

Di seguito sono riportati i risultati migliori ottenuti con ognuna delle tipologie di rete descritte nel capito Implementazione.

Dimensione vocabolario	%Accuracy
400	80.10
450	81.66
500	**83.33**
550	82.22

Tabella 6.3: Tuning dimensione vocabolario

I training delle reti neurali sfruttano la callback Keras *EarlyStopping*. Il parametro *patience* è settato a 5.

Prima di valutare il modello ed ottenere l'accuracy sul test test, vengono caricati i pesi della rete che durante la fase di training, ha ottenuto la loss sul validation set minore.

MLP

Il modello che ha riportato la minor loss sul validation set è il seguente:

Listing 6.1: Keras MLP

```
model = Sequential()
model.add(Dense(1024, input_shape=(vec_len,)))
model.add(Activation('relu'))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

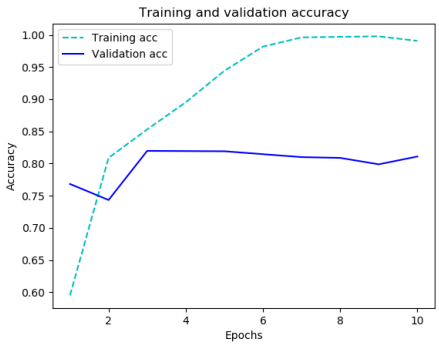
Accuracy: **82.51%**

Features set: **BOW + HL**

Epoca: **3**

In figura 6.1 si nota che alla terza epoca di allenamento si ottiene l'accuracy sul validation set maggiore, per poi ottenere un andamento pressoché costante. Per le epoche successive si denota un chiaro overfitting: la loss sul calcolata rispettto

al training set diminuisce e l'accuracy calcolata sul training set raggiunge già alla settima epoca il valore massimo.



(a) Accuracy

(b) Loss

Figura 6.1: Accuracy e Loss training MLP

CNN

Le reti convoluzionali utilizzate, hanno riportato i risultati migliori per gli input di un'unica categoria. Paragonando il loro utilizzo a quello della classificazione delle immagini, l'ipotesi è che l'aggiunta di feature concatenate porti allo stesso tipo di disturbo che la concatenazione e l'inserimento di elementi grafici (porzioni di immagini o singoli pixel) può verosimilmente implicare e quindi compromettere il riconoscimento da parte dei livelli convoluzionali, di caratteristiche dell'immagine originale.

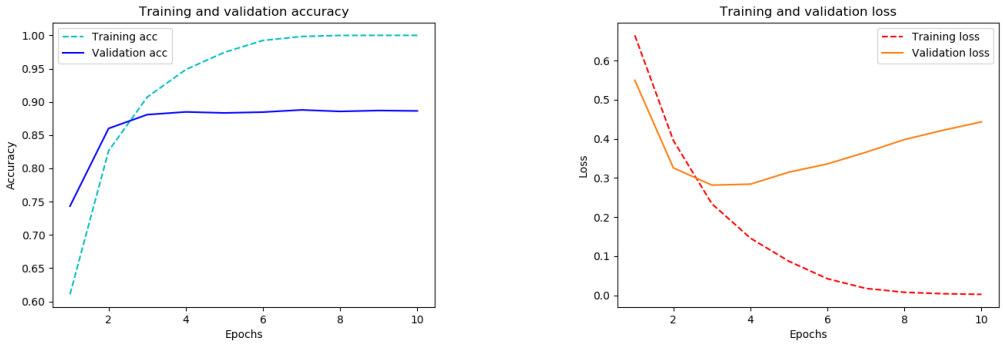
La rete che ha riportato i risultati migliori è quella che ha utilizzato l'embedding layer Keras.
Accuracy: **88.1%**
Features set: **POS Embedding word**
Epoche: **3**

modello:

Listing 6.2: Keras CNN

```
model = Sequential()
model.add(Embedding(input_dim = 32, embedding_lenght=64, input_length=500))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.20))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

In figura 6.2 si osserva un andamento dell'accuracy e della loss simile a quello ottenuto con il multiperceptron.



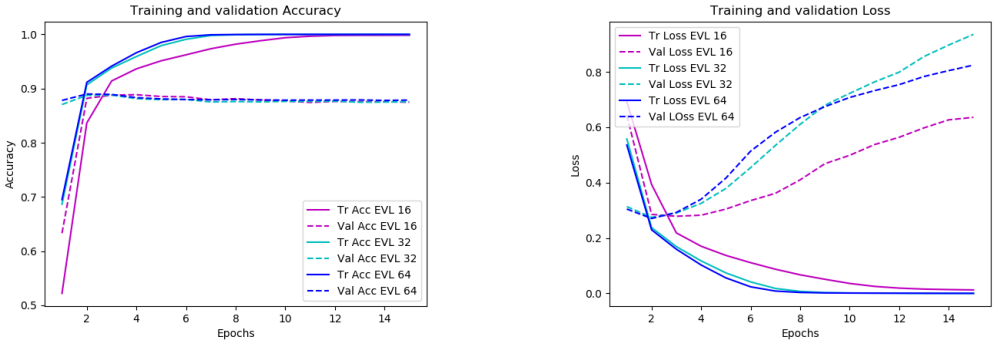
(a) Accuracy **(b)** Loss

Figura 6.2: Accuracy e Loss training CNN

La configurazione BOW, con le parole non arricchite dei pos tag ha ottenuto una accuracy pari a 87.13%.

Embedding size

Aumentando la lunghezza dei vettori embedding, da 16 a 64, oltre ad un maggior tempo di allenamento, non è stata riscontrata una differenza significativa. In figura 6.3 si osservano i corrispettivi andamenti di accuracy e loss, ottenuti con la precedente rete CNN. La grandezza di embedding scelta è **64**.

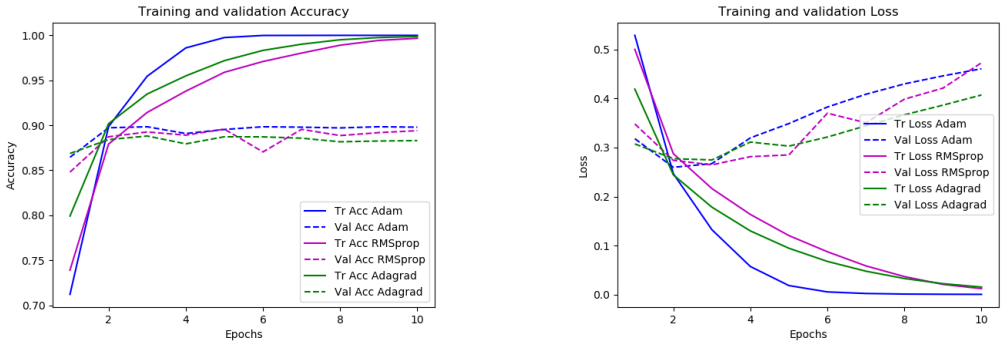


(a) Accuracy **(b)** Loss

Figura 6.3: Tuning embedding size con rete CNN

Ottimizzatori

In figura 6.4 sono riportati diversi andamenti di accuracy e loss ottenuti con la rete CNN utilizzando tre differenti ottimizzatori: Adam, RMSprop e Adagrad. L'ottimizzatore su cui si è fatto affidamento è Adam.



(a) Accuracy **(b)** Loss

Figura 6.4: Ottimizzatori - CNN

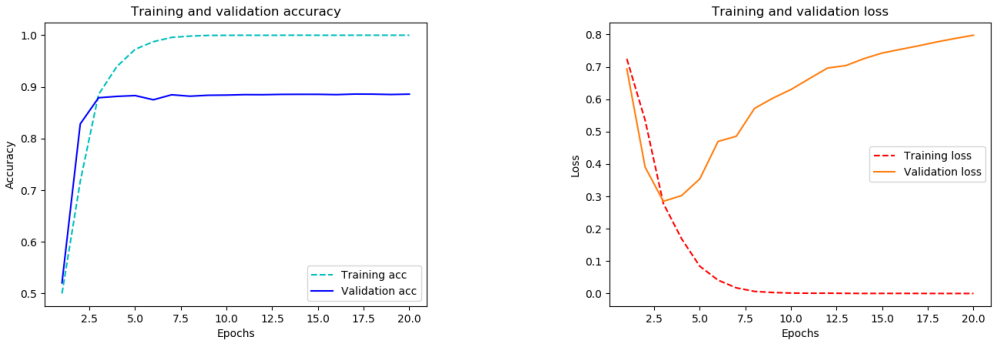
Modelli Concatenati

Accuracy: **90.45%**

Features set: **BOW + Tf-Idf + Embedding + HL**

Epoca: **3**

Quest'ultima architettura, è risultata la migliore. In figura 6.5 si osserva, similarmente ai modelli precedenti, il raggiungimento massimo della accuracy calcolata sul validation set, alla terza epoca di allenamento.



(a) Accuracy **(b)** Loss

Figura 6.5: Accuracy e Loss training Modelli concatenati

Riassunto Risultati Imdb

Nella tabella 6.4 sono riportati i risultati migliori in ordine decrescente di accuracy calcolata sul test set. Si conclude osservando che non è possibile determinare a priori quale feature set possa portare ai risultati migliori indifferentemente dal modello utilizzto.

Modello	Features set	Accuracy Test set(%)
Modelli Concatenati	**BOW + Tf-Idf + w2v + HL**	**90.45**
CNN	w2v(POS)	88.1
MLP	BOW + HL	85.5
SVM	POS	84.5
BernoulliNB	w2v	78.1

Tabella 6.4: Riassunto risultati Imdb Dataset

6.2 Twitter SemEval-2017

6.2.1 BernoulliNB

Features stack	Accuracy	Precision	Recall	F-score	AUC
BOW	0.757	0.74	0.76	0.75	0.798
Tf-Idf	0.749	0.74	0.75	0.75	0.789
w2v	0.615	0.64	0.62	0.63	0.602
POS	0.759	0.74	0.76	0.73	0.767
BOW + Tf-Idf	0.756	0.76	0.76	0.76	0.806
BOW + w2v	0.625	0.67	0.63	0.64	0.645
BOW + HL	**0.772**	**0.76**	**0.77**	**0.77**	**0.81**
BOW + Tf-Idf + w2v + POS	0.645	0.67	0.65	0.66	0.661

Tabella 6.5: Risultati Bernoulli - SemEval Dataset

6.2.2 SVM

Iper-parametri migliori ottenuti tramite GridSearchCV:

- C: 0.1
- kernel: lineare

Features stack	Accuracy	Precision	Recall	F-score	AUC
BOW	0.775	0.78	0.78	0.78	0.859
Tf-Idf	**0.786**	**0.79**	**0.79**	**0.79**	**0.868**
w2v	0.686	0.69	0.69	0.69	0.760
POS	0.779	0.78	0.78	0.78	0.863
BOW + Tf-Idf	0.723	0.72	0.72	0.72	0.805
BOW + w2v	0.773	0.77	0.77	0.77	0.855
BOW + HL	0.773	0.76	0.77	0.75	0.791
BOW + Tf-Idf + w2v + POS	0.722	0.71	0.71	0.71	0.715

Tabella 6.6: Risultati SVM - SemEval Dataset

6.2.3 Reti neurali

Per questo dominio, dato la dimensione ridotta del dataset, ogni tentativo di training tramite reti neurali non ha condotto ad alcun risultato superiore in accuracy a quelli ottenuti con SVC e BernoulliNB.

Conclusioni

Lo scopo di questo lavoro di tesi è stato lo studio della sentiment analysis a livello di frase, nello specifico la predizione della polarità attraverso l'apprendimento automatico supervisionato.

L'apprendimento è stato effettuato su due diversi domini: le recensioni di film ed i tweet. Per entrambi, si sono analizzate le diverse fasi di pre-processing suggerite dalla recente letteratura. Si è osservato quanto sia importante e necessario dover specializzare le fasi che precedono l'apprendimento, al dominio in analisi, cercando di andare oltre alle semplici statistiche riguardanti le frasi ed osservando ad un livello più alto le loro caratteristiche e soprattutto i metodi di scrittura degli utenti. Se per il dataset Imdb il dizionario dei token estratti è fortemente legato al dominio dei film, per i tweet, dati i numerosi topic diversi, un uso intensivo di abbreviazioni ed una alta percentuale di errori grammaticali, a parità di frasi fornite, il dizionario presenta un dizionario formato dal circa il doppio delle parole. Le principali problematiche affrontate in questa prima fase, hanno riguardato l'estrazione dei singoli token. Affinché si ottenessero sequence di token in grado di preservare la semantica delle frasi originali, sono stati testati diversi tokenizer. Il tokenizer ritenuto migliore è TreebankTokenizer.

I risultati riportati nel capitolo 6 hanno mostrato come la dimensione dei dataset abbia influenzato, come primo fattore, l'allenamento dei modelli di machine learning. Lo studio si è quindi soffermato maggiormente sul primo dataset, mostrando l'effettiva superiorità dell reti neurali nel classificare la polarità delle frasi. I modelli concatenati sono risultati essere il modello vincente.

Ulteriori difficoltà incontrate sono di carattere computazionale. Si sottolineano le lunghe fasi di training delle reti neurali effettuate su macchina locale, in quanto, prima di ottenere tuning dei parametri soddisfacenti, ogni epoca di allenamento, specialmente per le reti neurali convoluzionali, può presentare una durata media superiore alla decina di minuti. Nella seconda parte di sviluppo di tesi, i test sono stati eseguiti su una più performante macchina virtuale Amazon EC2, riducendo notevolmente i tempi di esecuzione delle fasi di training e valutazione.

6.3 Sviluppi futuri

Si vuole concludere questo lavoro con uno sguardo ai possibili studi e implementazioni future, al fine di aumentare l'accuratezza dei classificatori.

- Approfondire lo studio sui modelli concatenati, specializzando per ogni tipologia di feature, la miglior rete neurale che possa estrarre quante più caratteristiche importanti;
- Studiare più a fondo il ruolo, certamente importante, delle negazioni e degli intensificatori (*sentiment shifters*);
- Pre-classificare i messaggi in base alla soggettività, escludendo quelli non portatori di sentimenti o opinioni;
- Strettamento legato ai tweet, un possibile studio è basato sulle feature che non sono insite nel testo ma che discendono dalla rete sociale degli autori, come ad esempio i *follower* e i *following*;
- Per le recensioni dei film, poter analizzare la storia degli autori e confrontare sulla linea temporale i rating espliciti espressi in numero di stelle;
- Altro punto di notevole importanza è lo studio dell'ironia e del sarcasmo, task tra i più complicati nel campo della sentiment analysis.

Bibliografia

[1] Liu, Zang. Sentiment Analysis and Opinion Mining.
http://stp.lingfil.uu.se/~santinim/sais/2014/bingliu.pdf

[2] Jonathan G Fiscus and George R Doddington. Topic detection and tracking evaluation overview. pages 17-31. Springer, 2002.

[3] Emus, R., "Modeling and Representing Negation in Data-driven Machine Learning-based Sentiment Analysis", In Proceedings of the 1st International Workshop on Emotion and Sentiment in Social and Expressive Media (ESSEM),2013.

[4] Turney, P. (2002). Thumbs up or Thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of ACL-02, 40th Annual Meeting of the Association for Computational Linguistics.

[5] Diana, M., John.C., "Disambiguating Nouns, verbs, and adjectives using automatically acquired Selectional preferences", Association for Computational Linguistics , Vol. 29, 2003.

[6] Liu, X. Li, W.S. Lee, and P.S. Yu. Text classification by labeling words. In PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, pages 425 430. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[7] Melville, W. Gryc, and R.D. Lawrence. Sentiment analysis of blogs by combining lexical knowledge with text classification. In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1275-1284. ACM, 2009.

[8] S. Mahalakshmi, E. Sivasankar. Cross Domain Sentiment Analysis Using Different Machine Learning Techniques, in Proceedings of the Fifth International Conference on Fuzzy and Neuro Computing (FANCCO - 2015) pp 77-87

[9] Garg and Dan Roth, Understanding Probabilistic Classifiers Ashutosh

[10] IMDb dataset `http://ai.stanford.edu/~amaas/data/sentiment/`

[11] Treebank-3 `https://catalog.ldc.upenn.edu/LDC99T42`

[12] Bamber, 1975; Zweig e Campbell, 1993

[13] Piantadosi, Steven T. "Zipf's Word Frequency Law in Natural Language: A Critical Review and Future Directions." Psychonomic bulletin & review 21.5 (2014): 1112-1130. PMC. Web. 25 Aug. 2018.

[14] Imdb Dataset `http://ai.stanford.edu/~amaas/data/sentiment/`

[15] Svetlana Kiritchenko, Saif M. Mohammad, Examining Gender and Race Bias in Two Hundred Sentiment Analysis Systems. In Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM), New Orleans, USA, 2018.

[16] Hongxia Ma, Yangsen Zhang,Zhenlei Du. Cross-language sentiment classification based on Support Vector Machine. In 2015 11th International Conference on Natural Computation (ICNC)

[17] Sermanet P, LeCun Y. Traffic sign recognition with multi-scale convolutional networks. In Proceedings of the International Joint Conference on Neural Networks (IJCNN 2011), 2011.

[18] Elman J.L. Finding structure in time. Cognitive Science, 1990.

[19] Sepp Hochreiter and Jurger Schmidhuber. Long short-term memory. Neural Comput., 9(8):1735-1780, November 1997.

[20] Alex Graves, Generating Sequences With Recurrent Neural Networks, 2013

[21] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. World Wide Web Conference (WWW), 2003.

[22] Bo Pang and Lillian Lee. Thumbs up?: sentiment classification using machine learning techniques. Conference on Empirical Methods in Natural Language Processing (EMNLP), 10:86, 2002.

[23] Taboada M., Brooke J., Toloski M., Voll K., Stede M. 2011. Lexicon-based methods for sentiment analysis. In Association for Computational Linguistics, vol. 37, pp. 267-307

[24] Yoav Goldberg, Page 92, Neural Network Methods in Natural Language Processing, 2017.

[25] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. Journal of the American society for information science, 41(6):391, 1990.

[26] S. Das and M. Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. Asia Pacific Finance Association Annual Conference (APFA), 2001.

[27] Hang Cui, Vibhu Mittal, and Mayur Datar. Comparative experiments on sentiment classification for online product reviews. National Conference on Artificial Intelligence (AAAI), 21(2), 2006.

[28] Neha S. Joshi, Suhasini A. Itkat, "A Survey on Feature Level Sentiment Analysis" (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4), 2014, 5422-5425.

[29] WordNet - A Lexical Database for English `https://wordnet.princeton.edu/`

[30] Michael W. Berry, Soft Computing in Data Science, First International Conference, Scds 2015, Putrajaya, Malaysia, September 2-3, 2015

[31] Lei Zhang and Shuai Wang and Bing Liu, Deep Learning for Sentiment Analysis : A Survey, 2018. http://arxiv.org/abs/1801.07883

[32] Jeffrey Pennington and Richard Socher and Christopher D. Manning, Empirical Methods in Natural Language Processing (EMNLP), GloVe: Global Vectors for Word Representation, 2014 http://www.aclweb.org/anthology/D14-1162